Fuzzy Set Methods for Object Recognition in Space Applications

Second Quarter Report

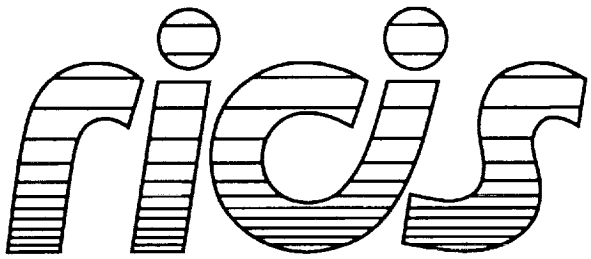
James M. Keller

University of Missouri-Columbia

10/1/91 - 12/31/91

Cooperative Agreement NCC 9-16
Research Activity No. SE.42

NASA Johnson Space Center
Information Systems Directorate
Information Technology Division



Research Institute for Computing and Information Systems
University of Houston-Clear Lake

INTERIM REPORT

The RICIS Concept

The University of Houston-Clear Lake established the Research Institute for Computing and Information Systems (RICIS) in 1986 to encourage the NASA Johnson Space Center (JSC) and local industry to actively support research in the computing and information sciences. As part of this endeavor, UHCL proposed a partnership with JSC to jointly define and manage an integrated program of research in advanced data processing technology needed for JSC's main missions, including administrative, engineering and science responsibilities. JSC agreed and entered into a continuing cooperative agreement with UHCL beginning in May 1986, to jointly plan and execute such research through RICIS. Additionally, under Cooperative Agreement NCC 9-16, computing and educational facilities are shared by the two institutions to conduct the research.

The UHCL/RICIS mission is to conduct, coordinate, and disseminate research and professional level education in computing and information systems to serve the needs of the government, industry, community and academia. RICIS combines resources of UHCL and its gateway affiliates to research and develop materials, prototypes and publications on topics of mutual interest to its sponsors and researchers. Within UHCL, the mission is being implemented through interdisciplinary involvement of faculty and students from each of the four schools: Business and Public Administration, Education, Human Sciences and Humanities, and Natural and Applied Sciences. RICIS also collaborates with industry in a companion program. This program is focused on serving the research and advanced development needs of industry.

Moreover, UHCL established relationships with other universities and research organizations, having common research interests, to provide additional sources of expertise to conduct needed research. For example, UHCL has entered into a special partnership with Texas A&M University to help oversee RICIS research and education programs, while other research organizations are involved via the "gateway" concept.

A major role of RICIS then is to find the best match of sponsors, researchers and research objectives to advance knowledge in the computing and information sciences. RICIS, working jointly with its sponsors, advises on research needs, recommends principals for conducting the research, provides technical and administrative support to coordinate the research and integrates technical results into the goals of UHCL, NASA/JSC and industry.

RICIS Preface

This research was conducted under auspices of the Research Institute for Computing and Information Systems by James M. Keller of the University of Missouri-Columbia. Dr. Terry Feagin was the initial RICIS research coordinator for this activity. Dr. A. Glen Houston, Director of RICIS and Assistant Professor of Computer Science, later assumed the research coordinator assignment.

Funding was provided by the Information Technology Division, Information Systems Directorate, NASA/JSC through Cooperative Agreement NCC 9-16 between the NASA Johnson Space Center and the University of Houston-Clear Lake. The NASA technical monitor for this activity was Robert N. Lea, of the Software Technology Branch, Information Technology Division, Information Systems Directorate, NASA/JSC.

Second Quarter Report: October 1, 1991 to December 31, 1991

Fuzzy Set Methods For Object Recognition In Space Applications

Fixed-Price Subcontract NO. 088

Under Cooperative Agreement No. NCC 9-16

Project No. SE. 42

To:

UNIVERSITY OF HOUSTON-CLEAR LAKE
2700 Bay Area Boulevard
Houston, TX 77058

UHCL Technical Representative:

Dr. Terry Feagin

Principal Investigator:

James M. Keller
Electrical and Computer Engineering Department
University of Missouri-Columbia
Columbia, Mo. 65211
(314) 882-7339
ecekeler@umcvmb.bitnet

Introduction

For the second quarter of this research contract, we are going to report progress on the following four Tasks (as described in the contract):

1. Fuzzy set-based decision making methodologies;

2. Feature Calculation;

3. Membership Calculation;

5. Acquisition of images.

Since there has been a delay in acquiring images from NASA, we have devoted more energies to tasks 1 and 2, and performed research on task 4: Clustering for curve and surface fitting (which was scheduled for the third quarter). The descriptions of our progress are written as "stand-alone" sections, including a copy of a manuscript on Quadric Shell Clustering which has been submitted to the 1992 IEEE Computer Vision/ Pattern Recognition Conference.

Also included is a Sun 4 tape (TAR format) including source code and images. The Read_Me file on the tape will describe its contents.

Fuzzy set-based decision making methodologies

We devoted most of our efforts this quarter to the development of the theory and application of methodologies for decision making under uncertainty. This section contains two subreports, the first on properties of general hybrid operators, while the second considers some new research on generalized threshold logic units.

HYBRID AGGREGATION OPERATORS

1.0 Introduction:

In this report, we explore the properties of the additive γ-model where the intersection part is first considered to be the product of the input values, and the union part is obtained by an extension of De Morgan's law to fuzzy sets. Then the Yager's class of union and intersection is used in the additive γ-model. The inputs are weighted to some power that represents their importance and thus their contribution to the compensation process.

2.0 Fuzzy Aggregation Connectives:

Fuzzy aggregation connectives are useful for aggregating memberships functions. The resulting membership depends on the type of aggregation connective used, and this type is dictated by the kind of attitude that we expect from this aggregation connective. These connectives are very useful in decision analysis and making. Several types of fuzzy connectives have been used:

2.1 The union connective:

It is used when the aggregated value is required to be high if any one of the input values ($x_i \in [0,1]$) is high. Examples are:

The maximum operator:

$$u(x_1, x_2, ..., x_m) = \max(x_1, x_2, ..., x_m) \quad (1)$$

Yager's union operator:

$$u(x_1, x_2, ..., x_m) = \min \{ 1, [\sum_{i=1}^{m} x_i p]^{1/p} \}, p \in [0,\infty). \quad (2)$$

2.2 The intersection connective:

It is used when the aggregated value is required to be high only when all the input values are high. Examples are:

The minimum operator:

$$i(x_1, x_2, ..., x_m) = \min(x_1, x_2, ..., x_m) \quad (3)$$

Yager's intersecion operator:

$$i(x_1, x_2, ..., x_m) = 1 - \min \{ 1, [\sum_{i=1}^{m} (1-x_i)^p]^{1/p} \}, p \in [0,\infty). \quad (4)$$

2.3 Mean operators:

Unlike the intersection and union operators, the mean operator does not take an extremist position in aggregating the input values, it rather regards the different criteria as mutually compensable in nature. It provides an aggregated value

$m(x_1, x_2, ..., x_m)$ such that

$$\min(x_1, x_2, ..., x_m) < m(x_1, x_2, ..., x_m) < \max(x_1, x_2, ..., x_m).$$

For example the generalized mean is defined by:

$$g(x_1, x_2, ..., x_m) = [\sum_{i=1}^{m} w_i x_i^p]^{1/p}, \text{ where } \sum_{i=1}^{m} w_i = 1. \quad (5)$$

The w_i are weights representing the importance of certain criteria, and $p \in (-\infty, \infty)$.

2.4 Compensatory or hybrid connectives:

In this type of connective, the high input values are allowed to compensate for the low ones. For example the additive and multiplicative γ operators are defined as weighted arithmetic / geometric means of union and intersection operators respectively:

$$A \oplus_\gamma B = (1 - \gamma) (A \cap B) + \gamma (A \cup B). \quad (6)$$

$$A \otimes_\gamma B = (A \cap B)^{(1 - \gamma)} (A \cup B)^\gamma. \quad (7)$$

It is clear that both of these operators can act as a pure intersection or union at the extremes: $\gamma = 0$ and 1 respectively. But they allow the intersection and union to compensate for each other when $0 \leq \gamma \leq 1$. Thus γ can be regarded as the parameter that controls the degree of compensation.

2.5 The multiplicative γ-model:

This model was introduced by Zimmerman and Zysno and is very similar to the multiplicative γ-operator:

$$y = \left(\prod_{i=1}^{m} x_i^{\delta_i}\right)^{(1-\gamma)} \left(1 - \prod_{i=1}^{m} (1 - x_i)^{\delta_i}\right)^{\gamma} \tag{8-a}$$

where

$$\sum_{i=1}^{m} \delta_i = m \tag{8-b}$$

and $0 \leq \gamma \leq 1$. (8-c)

The $x_i \in [0,1]$ are the inputs or criteria to be aggregated, δ_i represents the weight associated with the input x_i and is related to the importance of that input, and $\gamma \in [0,1]$ controls the degree of compensation between the union and intersection parts of the operator. Note that the intersection used in this case is the product of the inputs each weighted to some power δ_i :

$$y1 = \bigcap_{i=1}^{m} x_i^{\delta i} \tag{9}$$

$$y1 = \prod_{i=1}^{m} x_i^{\delta i}, \tag{10}$$

and the union is obtained from DeMorgan's law extended to fuzzy sets:

$$y2 = 1 - \bigcap_{i=1}^{m} (1 - x_i)^{\delta i} \tag{11}$$

$$y2 = 1 - \prod_{i=1}^{m} (1 - x_i)^{\delta i}. \tag{12}$$

But nothing restricts us from using other types of intersection and union connectives.

3.0 The Additive γ-model:

Similarly to the γ-model presented in 2.4, one can define an additive γ-model as:

$$y = (1 - \gamma)\, y_1 + \gamma\, y_2, \tag{13}$$

where y_1 and y_2 are an intersection and union as given by (9) and (11).

3.1 Using the product as an intersection:

In particular, using the product for the intersection as given by (10) and (12)

$$y = (1 - \gamma) \prod_{i=1}^{m} x_i \delta_i + \gamma \left(1 - \prod_{i=1}^{m} (1 - x_i)^{\delta_i}\right). \tag{14}$$

This model has several interesting properties that we proceed to show using the following partial derivatives:

$$\frac{\partial y}{\partial \gamma} = y_2 - y_1 \tag{15}$$

$$\frac{\partial y}{\partial x_k} = \delta_k \left\{ \frac{(1 - \gamma)\, y_1}{x_k} + \frac{\gamma\,(1 - y_2)}{(1 - x_k)} \right\} \tag{16}$$

Property 1. The sensitivity of y with respect to x_k is proportional to δ_k and is given by

$$S_{x_k} = \frac{x_k}{y} \frac{\partial y}{\partial x_k} = \frac{\delta_k}{y} \left\{ (1 - \gamma)\, y_1 + \frac{\gamma\, x_k\, (1 - y_2)}{(1 - x_k)} \right\} \tag{17}$$

Proof. This can be easily obtained from (16).

We can see that the contribution of x_k to the compensation process increases or decreases when the associated value of δ_k increases or decreases, because the value of the function inside the parentheses in (17) is always non-negative.

Property 2. The additive γ-model is a monotonically increasing function with respect to x_k.

Proof. This follows because the right hand side of (16) is always non-negative.

Property 3. The additive γ-model is a monotonically increasing function with respect to γ, and hence

$$y1 \le y \le y2. \tag{18}$$

Proof. It can be seen that $\prod_{i=1}^{m} (1 - x_i)^{\delta i} \le (1 - x_j)^{\delta j} \quad \forall\, j = 1,...,m.$

In particular, for that input x_* associated with the largest weight δ_{max}

$$\prod_{i=1}^{m} (1 - x_i)^{\delta i} \le (1 - x_*)^{\delta max} \le (1 - x_*).$$

The last inequality follows since $\delta_{max} \ge 1$ because of the constraint in (8-b). Therefore

$$y2 \ge x_*. \tag{19-a}$$

Similarly,

$\prod_{i=1}^{m} x_i^{\delta i} \le x_j^{\delta j} \quad \forall\, j = 1,...,m.$ In particular,

$\prod_{i=1}^{m} x_i^{\delta i} \le x_*^{\delta max} \le x_*.$ Therefore

$$y1 \le x_*. \tag{19-b}$$

It follows from (19) that $y2 \ge y1$. Hence using (15) $\frac{\partial y}{\partial \gamma} \ge 0$.

Propety 4. The range of the additive γ-model is as follows:

$$x_{min}^{m} \le y \le 1 - (1 - x_{max})^{m}, \tag{20}$$

where

$$x_{min} = \min(x_1, x_2, ..., x_m)$$

and

$$x_{max} = \max(x_1, x_2, ..., x_m).$$

Proof.

$$\prod_{i=1}^{m} x_i^{\delta i} \geq \prod_{i=1}^{m} x_{min}^{\delta i} = x_{min}^{m}, \text{ hence}$$

$$y1 \geq x_{min}^{m}. \qquad (21\text{-a})$$

Similarly,

$$\prod_{i=1}^{m} (1 - x_i)^{\delta i} \geq \prod_{i=1}^{m} (1 - x_{max})^{\delta i} = (1 - x_{max})^{m}, \text{ hence}$$

$$y2 \leq 1 - (1 - x_{max})^{m}. \qquad (21\text{-b})$$

Finally, the range is established as in (20) using (18) and (21).

Therefore the range of the additive γ-model is limited and does not extend to 0 and 1 unlike the Yager's union and intersection which can be parametrized to do so. However, if m is sufficiently large, the range of the additive γ-model may still suffice for most applications. One way to enlarge the range while preserving all the properties is to loosen the constraint in (8-b), and replace it by the following constraint:

There exists at least one $\delta_i \geq 1$. (21-c)

This constraint is necessary in order to preserve property 3. In this case, the range becomes

$$x_{min}^{(\sum_{i=1}^{m}\delta_i)} \leq y \leq 1 - (1 - x_{max})^{(\sum_{i=1}^{m}\delta_i)}, \qquad (21\text{-d})$$

which gets closer to [0,1] as $\sum_{i=1}^{m}\delta_i$ increases

4.0 The Additive γ-model with Yager's union and intersection:

In (13), we can use Yager's intersection and union. Therefore y_1 is given by (4) and y_2 is given by (2):

$$y_1 = 1 - \min \{1, f_1(x_i, p)\}, \tag{22-a}$$

where

$$f_1(x_i, p) = \left[\sum_{i=1}^{m} (1-x_i)^p \right]^{1/p}, \; p \in [0,\infty), \tag{22-b}$$

and

$$y_2 = \min \{1, f_2(x_i, p)\}, \tag{23-a}$$

where

$$f_2(x_i, p) = \left[\sum_{i=1}^{m} x_i^p \right]^{1/p}, \; p \in [0,\infty). \tag{23-b}$$

Note that

$$f_1(x_i, p) = f_2(1 - x_i, p) \tag{23-c}$$

4.1 Properties of the Yager's union and intersection connectives:

Property 1. y_1 is monotonically non decreasing, and y_2 is monotonically non increasing with respect to p.

Proof. Consider

$$\ln f_2 = 1/p \ln \left[\sum_{i=1}^{m} x_i^p \right]$$

$$\frac{\partial \ln f_2}{\partial p} = \frac{1}{p^2}\left[p \frac{\sum_{i=1}^{m} x_i^p \ln x_i}{\sum_{i=1}^{m} x_i^p} - \ln \sum_{i=1}^{m} x_i^p \right]$$

$$\frac{\partial \ln f_2}{\partial p} = \frac{1}{p^2 \sum_{i=1}^{m} x_i^p} \sum_{i=1}^{m} \left[x_i^p \ln \left(\frac{x_i^p}{\sum_{i=1}^{m} x_i^p} \right) \right] \leq 0$$

since $\frac{x_i^p}{\sum_{i=1}^{m} x_i^p} \leq 1.$

Therefore $f2(x_i, p)$ is monotonically non increasing with respect to p. Thus y_2 is monotonically non increasing with respect to p. It follows from (23-c) that $f_1(x_i, p)$ is monotonically non increasing with respect to p, and hence y_1 is monotonically non decreasing with respect to p.

Property 2.

a. $\lim_{p \to 0} y_1 = i_{min}(x_1, x_2, ..., x_m) = \begin{cases} x_k & \text{when } x_i = 1 \ \forall \ i \neq k \\ 0 & \text{otherwise} \end{cases}$

b. $\lim_{p \to \infty} y_1 = \min(x_1, x_2, ..., x_m)$

c. $\lim_{p \to 0} y_2 = u_{max}(x_1, x_2, ..., x_m) = \begin{cases} x_k & \text{when } x_i = 0 \ \forall \ i \neq k \\ 1 & \text{otherwise} \end{cases}$

d. $\lim_{p \to \infty} y_2 = \max(x_1, x_2, ..., x_m)$

Proof.

a. Suppose there exists only one input $x_k \neq 1$ while all the other inputs $x_i = 1 \ \forall \ i \neq k$. Hence

$$f_1(x_i, p) = [\,(1-x_k)^p\,]^{1/p} = (1-x_k) \le 1.$$

Therefore from (22-a), $y_1 = x_k$.

On the other hand suppose that inputs $x_i \neq 1$ for i = 1,...,s, while the rest of the inputs $x_i = 1$ for i = s+1,...,m. Hence

$$\lim_{p\to 0} \ln f_1(x_i, p) = \lim_{p\to 0} (1/p) \ln [\sum_{i=1}^{s} (1-x_i)^p\,] = \lim_{p\to 0} (1/p) \ln s = \infty.$$

It follows from (22-a) that $\lim_{p\to 0} y_1 = 0$.

b. Since

$\lim_{p\to\infty} f_2(x_i, p) = \max(x_1, x_2, ..., x_m)$ as will be proved in in **d**, it follows from (23-c) that

$$\lim_{p\to\infty} f_1(x_i, p) = \max(1 - x_1, 1 - x_2, ..., 1 - x_m)$$

$$= 1 - \min(x_1, x_2, ..., x_m) \le 1.$$

Hence property 2 b. follows directly from (22-a).

c. Suppose there exists only one input $x_k \neq 0$ while all the other inputs $x_i = 0\ \forall\ i \neq k$.

Hence

$$f_2(x_i, p) = [\,x_k^p\,]^{1/p} = x_k \le 1.$$

Therefore from (23-a), $y_2 = x_k$.

On the other hand suppose that inputs $x_i \neq 0$ for i = 1,...,s, while the rest of the inputs $x_i = 0$ for i = s+1,...,m. Hence

$$\lim_{p\to 0} \ln f_2(x_i, p) = \lim_{p\to 0} (1/p) \ln [\sum_{i=1}^{s} x_i^p\,] = \lim_{p\to 0} (1/p) \ln s = \infty.$$

It follows from (23-a) that $\lim_{p\to 0} y_2 = 1$.

d. $\lim_{p\to\infty} \ln(f_2(x_i, p)) = \lim_{p\to\infty} \dfrac{\ln(\sum_{i=1}^{m} x_i^p)}{p}$

$$= \lim_{p\to\infty} \frac{\sum_{i=1}^{m} x_i^p \ln x_i}{\sum_{i=1}^{m} x_i^p} \text{ using L'Hopital's rule,}$$

$$= \lim_{p\to\infty} \sum_{i=1}^{m} \frac{x_i^p}{\sum_{i=1}^{m} x_i^p} \ln x_i$$

$$= \lim_{p\to\infty} \sum_{i=1}^{m} \frac{\left(\frac{x_i}{x_k}\right)^p}{\sum_{i=1}^{m} \left(\frac{x_i}{x_k}\right)^p} \ln x_i, \text{ where } x_k =$$

$\max(x_1,x_2,\ldots,x_m)$

$$= \lim_{p\to\infty} \sum_{i=1}^{m} \frac{\left(\frac{x_i}{x_k}\right)^p}{1 + \sum_{\substack{i=1 \\ i\neq k}}^{m} \left(\frac{x_i}{x_k}\right)^p} \ln x_i$$

$$= \lim_{p\to\infty} \sum_{i=1}^{m} \left(\frac{x_i}{x_k}\right)^p \ln x_i$$

$$= \lim_{p\to\infty} \left(\ln x_k + \sum_{\substack{i=1 \\ i\neq k}}^{m} \left(\frac{x_i}{x_k}\right)^p \ln x_i \right)$$

$$= \ln x_k.$$

Therefore

$$\lim_{p \to \infty} f_2(x_i, p) = x_k = \max(x_1, x_2, ..., x_m) \leq 1.$$

Hence property 2 d. follows directly from (23-a).

Property 3. y_1 and y_2 are monotonically non decreasing with respect to x_k.

Proof.

$$\frac{\partial f_2(x_i, p)}{\partial x_k} = x_k^{p-1} \left[\sum_{i=1}^{m} x_i^p \right]^{1/p-1} \geq 0.$$

Therefore $f_2(x_i, p)$ and hence y_2 is monotonically non decreasing with respect to x_k. It follows from (23-c) that $f_1(x_i, p)$ is monotonically non increasing and therefore y_1 as given in (22-a) is monotonically non decreasing with respect to x_k.

Property 4. The range of y_1 and y_2 is as follows

$$\min(x_1, x_2, ..., x_m) \leq y_1 \leq \min(x_1, x_2, ..., x_m) \qquad (24\text{-a})$$

$$\max(x_1, x_2, ..., x_m) \leq y_2 \leq \max(x_1, x_2, ..., x_m) \qquad (24\text{-b})$$

Proof. This property follows directly from properties 1 and 2.

Therefore both y_1 and y_2 can be tuned to act as an intersection and union respectively with the desired attitude, from the least optimistic to the most optimistic operator, by a proper selection of the parameter p. Property 4 also guarantees that the union is always greater than the intersection even when different values of p are used in y_1 and y_2.

$$y_2 \geq y_1. \qquad (24\text{-c})$$

4.2 The additive γ-model with Yager's union and intersection and weighted inputs.

The output value has the form

$$y = (1 - \gamma)\, y_1 + \gamma\, y_2, \tag{25}$$

where

$$y_1 = 1 - \min\{1, f_1(x_i^{\delta i}, p)\}, \tag{26-a}$$

where

$$f_1(x_i^{\delta i}, p) = \Big[\sum_{i=1}^{m} (1 - x_i^{\delta i})^p \Big]^{1/p},\ p \in [0,\infty), \tag{26-b}$$

and

$$y_2 = \min\{1, f_2(x_i^{\delta i}, p)\}, \tag{27-a}$$

where

$$f_2(x_i^{\delta i}, p) = \Big[\sum_{i=1}^{m} (x_i^{\delta i})^p \Big]^{1/p},\ p \in [0,\infty). \tag{27-b}$$

Note that

$$f_1(x_i^{\delta i}, p) = f_2(1 - x_i^{\delta i}, p) \tag{27-c}$$

where

$$\sum_{i=1}^{m} \delta_i = m \tag{27-d}$$

and $0 \le \gamma \le 1$. (27-e)

The $x_i \in [0,1]$ are the inputs or criteria to be aggregated, δ_i represents the weight associated with the input x_i and is related to the importance of that input, and $\gamma \in [0,1]$ controls the degree of compensation between the union and intersection parts of the operator.

It is easy to see that y_1 and y_2 still satisfy all the properties shown in 4.1, if we replace the inputs x_i by their weighted version

$a_i = x_i^{\delta i}$.

This model has several interesting properties that we proceed to show using the following partial derivatives:

$$\frac{\partial y}{\partial \gamma} = y2 - y1 \tag{28}$$

$$\frac{\partial y}{\partial x_k} = \frac{\partial y}{\partial a_k}\frac{\partial a_k}{\partial x_k} = \delta_k x_k^{\delta k-1}\{(1-\gamma)\frac{\partial y1}{\partial a_k} + \gamma\frac{\partial y2}{\partial a_k}\} \tag{29}$$

Property 1. The sensitivity of y with respect to x_k is proportional to δ_k and is given by

$$S_{x_k} = \frac{x_k}{y}\frac{\partial y}{\partial x_k} = \frac{\delta_k a_k}{y}\{(1-\gamma)\frac{\partial y1}{\partial a_k} + \gamma\frac{\partial y2}{\partial a_k}\} \tag{30}$$

Proof. This can be easily obtained from (29).

We can see that the contribution of x_k to the compensation process increases or decreases when the associated value of δ_k increases or decreases, because the value of the function inside the parentheses in (30) is always non-negative.

Property 2. The additive γ-model using Yager's union and intersection is a monotonically increasing function with respect to x_k.

Proof. This follows because the right hand side of (29) is always non-negative from property 3 in 4.1 if x_{l_k} is replaced by a_k.

Property 3. The additive γ-model using Yager's union and intersection is a monotonically increasing function with respect to γ, and hence

$$y1 \le y \le y2. \tag{31}$$

Proof. This property follows directly from (28) and (24-c).

Propety 4. The range of the additive γ-model using Yager's union and intersection is as follows:

$$i_{min}(x_1^{\delta 1}, x_2^{\delta 2}, ..., x_m^{\delta m}) \le y \le u_{max}(x_1^{\delta 1}, x_2^{\delta 2}, ..., x_m^{\delta m}), \tag{32-a}$$

where

$$i_{min}(x_1^{\delta 1}, x_2^{\delta 2}, ..., x_m^{\delta m}) = \begin{cases} x_k^{\delta k} & \text{when } x_i^{\delta i} = 1 \ \forall \ i \neq k \\ 0 & \text{otherwise} \end{cases} \tag{32-b}$$

$$u_{max}(x_1^{\delta 1}, x_2^{\delta 2}, ..., x_m^{\delta m}) = \begin{cases} x_k^{\delta k} & \text{when } x_i^{\delta i} = 0 \ \forall \ i \neq k \\ 1 & \text{otherwise} \end{cases} \tag{32-c}$$

Proof. This property follows directly from (24-a), (24-b) and (31).

Therefore the range of the additive γ-model using Yager's union and intersection is not as limited as the additive γ-model using the product as an intersection, and does extend to 0 and 1 depending on the choice of the parameter p.

Property 5. In the case where the constraint on the δ_i is loosened as in (21-c), all the previous properties still hold. In addition, the additive γ-model using Yager's union and intersection is a monotonically non increasing function with respect to δ_k.

Proof.

$$\frac{\partial y}{\partial \delta_k} = \frac{\partial y}{\partial a_k} \frac{\partial a_k}{\partial \delta_k} = a_k \ln x_k \{(1 - \gamma) \frac{\partial y1}{\partial a_k} + \gamma \frac{\partial y2}{\partial a_k} \} \tag{33}$$

The expression on the right hand side is obviously negative since $x_i \in [0,1]$ and the quantity inside the brackets is positive.

References

1. R. C. Luo and M. G. Kay, "Multisensor Integration and Fusion in Intelligent Systems", *IEEE Transactions on Systems Man and Cybernetics*, Vol. 19, No. 5, Sept/Oct. 1989, pp. 901-931.

2. J. M. Richardson and K. A. Marsh, "Fusion of Multisensor Data", *International Journal of Robotics Research*, Vol. 7, No. 6, 1988, pp. 78-96.

3. R. Krishnapuram and J. Lee, "Fuzzy-Connective-Based Hierarchical Aggregation Networks for Decision Making", *Fuzzy Sets and Systems*, Vol. 46, 1992, pp. 1-17.

4. R. Krishnapuram and J. Lee, "Fuzzy-Compensative-Connective-Based Hierarchical Networks and their Application to Computer Vision" submitted to the *Journal of Neural Networks*.

5. H. Tahani, and J. Keller, "Automated Calculation of Non-additive Measures for Object Recognition", *Proceedings of the SPIE Symposium on Intelligent Robots and Computer Vision IX*, Boston, MA, November, 1990.

6. H. Tahani, "The Generalized Fuzzy Integral in Computer Vision", Ph.D. Dissertation, University of Missouri-Columbia (J. Keller, advisor), 1991.

7. D. Dubois and H. Prade, "A Review of Fuzzy Set Aggregation Connectives", *Information Sciences*, vol. 36, no. 1&2, pp. 85-121, July/August 1985.

8. M. Mizumoto, "Pictorial Representations of Fuzzy Connectives, Part I: Cases of t-norms, t-conorms, and Averaging Operators", *Fuzzy Sets and Systems*, Vol. 31, 1989, pp. 217-242.

9. M. Mizumoto, "Pictorial Representations of Fuzzy Connectives, Part II: Cases of Compensatory Operators, and Self-Dual Operators", *Fuzzy Sets and Systems*, Vol. 32, 1989, pp. 45-79.

10. H. Dyckhoff and W. Pedrycz, "Generalized Means as a Model of Compensation Connectives", *Fuzzy Sets and Systems*, vol. 14, no. 2, pp. 143-154, November 1984.

11. H.-J. Zimmermann and P. Zysno, "Decisions and Evaluations by Hierarchical Aggregation of Information", *Fuzzy Sets and Systems*, vol. 10, no. 3, pp. 243-260, July 1983.

12. E. M. Riseman and A. R. Hanson, "A Methodology for the Development of General Knowledge-Based Vision Systems", in *Vision Systems and Cooperative Computation*, M. A. Arbib (Editor), MIT Press, Cambridge Ma, 1988.

13. T. L. Huntsburger, C. L. Jacobs and R. C. Cannon, "Iterative Fuzzy Image Segmentation", *Pattern Recognition*, Vol. 18, 1985, pp. 131-138.

14. J. Bezdek, R. Canon, R. Dave, M. Trivedi, "Segmentation of a Thematic Mapper Image Using the Fuzzy C-Means Clustering Algorithm", *IEEE Transactions on Geo. Science and Remote Sensing*, Vol. GE 24, No. 3, 1986, pp. 400-408.

15. J. Keller, and J. Givens, "Membership Function Issues in Fuzzy Pattern Recognition," *Proceedings, International Conference on Systems, Man, Cybernetics*, Tucson, AZ, November 1985, pp. 210-214.

16. K. Unklesbay, J. Keller, N. Unklesbay and D. Subhangkasen, "Determination of Doneness of Beef Steaks Using Fuzzy Pattern Recognition", *Journal of Food Engineering*, Vol. 8, No. 2, 1989, pp. 79-90.

17. J. Keller and R. Krishnapuram, "Fuzzy Set Methods in Computer Vision", *An Introduction to Fuzzy Logic Applications in Intelligent Systems*, R. Yager and L. Zadeh (eds)., to appear.

18. J. C. Bezdek: *Pattern Recognition with Fuzzy Objective Function Algorithms*, Plenum Press, 1981.

19. I. Gath and A. B. Geva, Unsupervised optimal fuzzy clustering, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 11, July 1989, 773-781.

20. R. Krishnapuram and A. Munshi, "Cluster-Based Segmentation of Range Images Using Differential-Geometric Features", under review.

21. Krishnapuram and C.-P. Freg, "Fitting an Unknown Number of Lines and Planes to Image Data through Compatible Cluster Merging", under revision for *Pattern Recognition.*

22. R. Krishnapuram and Chih-Pin Freg, "Fuzzy Algorithms to Find Linear and Planar Clusters and Their Applications", accepted for presentation at the *IEEE International Conference on Computer Vision and Pattern Recognition,* Hawaii, June 1991.

23. J. Keller and D. Hunt, "Incorporating Fuzzy Membership Functions into the Perceptron Algorithm," *IEEE Transactions on Pattern Analysis Machine Intelligence*, Vol. PAMI-7, No. 6, November, 1985, pp. 693-699.

24. R. P. Yager, S. Ovchinnikov, R. M. Tong, N. T. Nguen, *Fuzzy Sets and Applications: Selected Papers by L. A. Zadeh*, John Wiley, New York, 1987.

25. J. Keller, D. Subhangkasen, K. Unklesbay, N. Unklesbay, "Approximate reasoning for Recognition in Color Images of Beef Steaks", to appear in the *International Journal of General Systems*, Special Issue on Pattern Recognition, 1990.

26. J. Keller, M. Gray and J. Givens, "A Fuzzy K Nearest Neighbor Algorithm", *IEEE Transactions on Systems, Man and Cybernetics*, Vol. SMC-15, No. 4, July/Aug. 1985, pp. 580-585.

27. J. Keller and H. Qiu, "Fuzzy Set Methods in Pattern Recognition," in *Pattern Recognition*, Lecture Notes in Computer Science, Vol. 301, J. Kittler, (ed.), Springer-Verlag, Berlin, 1988, pp. 173-182.

28. J. Keller, G. Hobson, J. Wootton, A. Nafarieh and K. Luetkemeyer, "Fuzzy Confidence Measures in Midlevel Vision," *IEEE Transactions on Systems, Man and Cybernetics*, Vol. SMC-17, No. 4, 1987, pp. 676-683.

29. A. Nafarieh, and J. Keller, "A Fuzzy Logic Rule-Based Automatic Target Recognizer", *International Journal of Intelligent Systems*, to appear, 1991.

30. P. J. Besl and R. C. Jain, "Segmentation through variable-order surface fitting", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 10, March 1988, pp. 167-192.

31. N. Yokoya and M. D. Levine, "Range Image Segmentation based on differential Geometry: A Hybrid Approach",*IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 11, June 1989, pp. 643-649.

32. A. Pentland, "Fractal-Based Description of Natural Scenes", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 6, No. 6, 1984, pp. 661-674.

33. J. Keller, R. Crownover, and R. Chen, "Characteristics of Natural Scenes Related to the Fractal Dimension," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. PAMI-9, No. 4, 1987, pp. 621-629.

34. J. Keller, S. Chen and R. Crownover, "Texture Description Through Fractal Geometry," *Computer Vision, Graphics and Image Processing*, vol. 45, 1989, pp. 150-166.

35. S. Chen, J. Keller, and R. Crownover, "Shape from Fractal Geometry", *Artificial Intelligence*, Vol. 43, 1990, pp. 199-218.

36. B.B. Mandelbrot and J. Van Ness, "Fractional Brownian motions, fractional noises and applications," *SIAM Rev.*, vol. 10, no. 4, p. 422, 1968.

37. B. B. Mandelbrot, *The Fractal Geometry of Nature*, Freeman, San Francisco, 1983.

38. R. Voss, "Random Fractals: Characterization and Measurement", in *Scaling Phenomenon in Disordered Systems*, R. Pynn and A. Skjeltorp Eds., Plenum Press, New York, 1986.

39. J. Keller and Young-Bo Seo, "Local Fractal Geometric Features for Image Segmentation", to appear in the *International Journal of Imaging Systems and Technology*, Special Issue on Computer Vision, 1990.

40. R. M. Bolle and B. Vemuri, "On Three-Dimensional Surface Reconstruction Methods", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 13, No. 1, January 1991, pp. 1-13.

41. D. E. Gustafson and W. Kessel, Fuzzy clustering with a fuzzy covariance matrix, *Proceedings of IEEE-CDC*, **2** (K. S. Fu ed.), IEEE Press, N. J., p. 761 (1979).

42. E. Diday and J. C. Simon, Clustering analysis, in *Digital Pattern Recognition*, K. S. Fu (ed.), Springer, New York, p. 47 (1976).

43. O. D. Faugeras and M. Hebert, "The representation, recognition, and positioning of 3-D shapes from range data", In *Three Dimensional Machine Vision*, T. Kanade (Ed.), Kluwer, Norwell, Ma, 1987.

44. D. Casasent and R. Krishnapuram, "Curved Object Location by Hough Transformations and Inversions", *Pattern Recognition*, Vol.20, No. 2, 1987, pp. 181-188.

45. R. Krishnapuram and D. Casasent, "Determination of Three-Dimensional Object Location and Orientation from Range Images", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 11, No. 11, November 1989, pp. 1158-1167.

46. J. Keller, H. Qiu, and H. Tahani, "The Fuzzy Integral in Image Segmentation," *Proceedings, NAFIPS-86*, New Orleans, June 1986, pp. 324-338.

47. S.K. Pal and A. Rosenfeld, "Image Enhancement and Thresholding by Optimization of Fuzzy compactness", *Pattern Recognition Letters*, Vol. 7, 1988, pp. 77-86.

48. J. Keller and L. Stzandera, "Spatial Relationships Among Fuzzy Subsets of an Image", *Proc. Int. Symposium in Uncertainty Modeling and Analysis*, Univ. of Maryland, December, 1990, pp. 207-211.

49. L. Zadeh, "The concept of a linguistic variable and its application to approximate reasoning", *Information Sciences*, Part 1, Vol. 8, pp. 199-249; Part 2, Vol. 8, pp. 301-357; Part 3, Vol. 9, 43-80, 1975.

50. J. Keller and H. Tahani, "Implementation of Conjunctive and Disjunctive Fuzzy Logic Rules with Neural Networks", *International Journal of Approximate Reasoning*, Special issue on "Fuzzy Logic and Neural Networks for Control", to appear, 1991.

51. A. Rosenfeld and A.C. Kak, *Digital Picture Processing*, Vol. 2, Academic Press, N.Y., 1982.

52. D. Dubois and H. Prade, A class of fuzzy measures based on triangular norms, *Internat. J. Gen. Systems* **8** (1982) 43-61.

53. G.J. Klir and T.A Folger, *Fuzzy sets, Uncertainty and information* (Prentice Hall, Englewood Cliffs, NJ, 1988).

54. R.J Krishnapuram, A belief maintenance scheme for hierarchical knowledge-based image analysis systems, in: *Proc. 3rd IFSA Congress*, Seattle, 333-336.

55. D. Sabbah, Computing with connections in visual recognition of origami objects, *Cognitive Sci.* **9** (1985) 25-50.

56. U. Thole and H.-J. Zimmerman, On the suitability of minimum and product operations for the intersection of fuzzy sets, *Fuzzy sets and systems* **2**, (1979) 167-180.

57. R.R Yager, Fuzzy decision making including unequal objectives, *Fuzzy sets and systems* **1** (1978) 87-95.

58. R.R Yager, On ordered weighted averaging aggregation operators in multicriteria decision making, *IEEE Trans. Systems Man Cybernet.* **18** (1988) 183-190.

59. H.-J. Zimmerman and P. Zysno, Latent connectives in human decision making, *Fuzzy Sets and Systems* **4** (1980) 37-51.

GENERALIZED THRESHOLD LOGIC UNITS

1. Introduction

Finite automata first emerged as a model of neural networks in the work of McCulloch and Pitts (1943) and are natural extensions of switching circuits. Switching circuits consisting of conventional gates (developed from boolean logic) can require many components and interconnections whereas another type of switching device called the threshold logic unit (developed from threshold logic), usually does not. There are systematic methods to synthesize networks consisting of threshold logic units, and these methods can be used to synthesize decision networks when the input and output variables are binary. This method of synthesis avoids the problems due to local minima in other network training schemes such as the back propagation algorithm. Extension of binary logic synthesis methods to multiple valued logic synthesis methods will enable us to synthesize decision networks when the input/output variables are not binary. This will be discussed in section 7. We now discuss the idea behind the threshold logic unit and its applications to pattern classification .

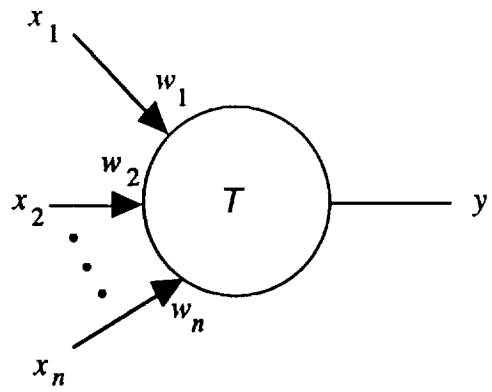
2. Threshold logic unit

A threshold unit (gate) consists of n two valued inputs $x_1, \ldots, x_n$ and a single two valued output y. Its internal parameters are a threshold T and weights $w_1, \ldots, w_n$, where each weight is associated with a particular input variable x_i. The values of threshold T and the weights w_i (i=1, . . .,n) may be any real, finite, positive or negative numbers. The input-output relation of a threshold logic element is defined as follows:

$$y = 1 \quad \text{if and only if} \quad \sum_{i=1}^{n} w_i x_i \geq T$$

$$y = 0 \quad \text{if and only if} \quad \sum_{i=1}^{n} w_i x_i < T$$

where the sum and product operations are the conventional arithmetic ones. The sum $\sum_{i=1}^{n} w_i x_i$ is called the weighted sum of the element. The symbolic representation of the threshold logic unit is shown in the figure below.



3. Application to pattern recognition

The main purpose of a pattern recognition system is to make decisions concerning class membership. In the two-class classifying problem, as well as in the multiclass problem, an equation of a surface that separates the pattern classes is of great interest. If the surface is hyperplanar, then we will call this equation a hyperplane equation (or decision function) $d[X]$, and it can be represented as follows:

$$d[X]=w_1x_1 + \cdots + w_nx_n = T$$

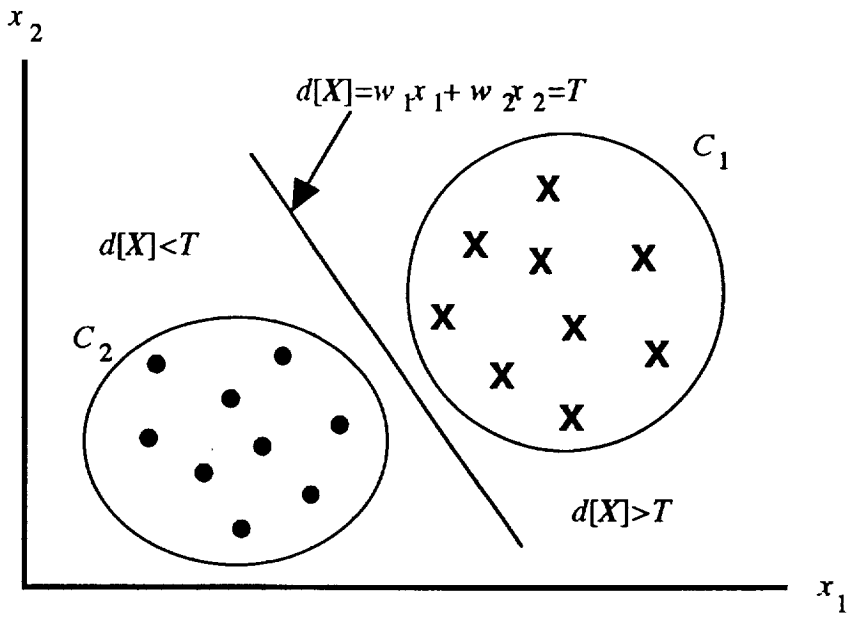
$$\text{or } d[X] = \sum_{i=1}^{n} w_i x_i = T$$

where $X = (x_1,\ldots,x_n)^t$ is called a pattern vector and x_i is the value of the i^{th} feature,

$w_i = i^{th}$ weight,

and T = threshold.

As an illustration, for the two-class two-feature case, for values of X that make $d[X] \geq T$, we can consider X belonging to one class, and for $d[X] < 0$, we can consider X belonging to another. The two pattern populations can be separated by the linear equation $d[X] = w_1x_1 + w_2x_2 = T$ and is shown below.



From the figure, it is clear that for values of x_1 and x_2 that make $d[X] \geq T$, we can consider X belonging to C_1, and to C_2 if $d[X] < T$. This is in the form of the threshold unit discussed previously and the next step is to determine the parameters w and T. Methods for obtaining these parameters are discussed in the next section.

4. Method of obtaining the hyperplane equation via threshold logic

The method of obtaining the hyperplane equation that separates different classes can be illustrated by the following two-class three-feature problem. Suppose we have 8 patterns each with three features with 4 patterns classified as class 0 and 4 patterns classified as class 1. Let the following truth table describe the problem.

Features			Class	Hyperplane equation
x_1	x_2	x_3	output d	$w_1x_1+w_2x_2+w_3x_3=T$
0	0	0	0	$0 < T$
0	0	1	0	$w_3 < T$
0	1	0	0	$w_2 < T$
0	1	1	1	$w_2+w_3 \geq T$
1	0	0	0	$w_1 < T$
1	0	1	1	$w_1 +w_3 \geq T$
1	1	0	1	$w_1 +w_2 \geq T$
1	1	1	1	$w_1 +w_2 + w_3 \geq T$

From the above truth table, for the output to be class 0 (d=0), four inequalities yield

$0<T$

$w_3<T$

$w_2<T$

$w_1<T$

and for the output to be class 1 (d=1), four inequalities yields

$w_2+w_3\geq T$

$w_1+w_3\geq T$

$w_1+w_2\geq T$

$w_1+w_2+w_3\geq T$

Combining these inequalities, we have

$0<T$

$w_1\geq 0$

$w_2\geq 0$

$w_3\geq 0$

and finally the condition,

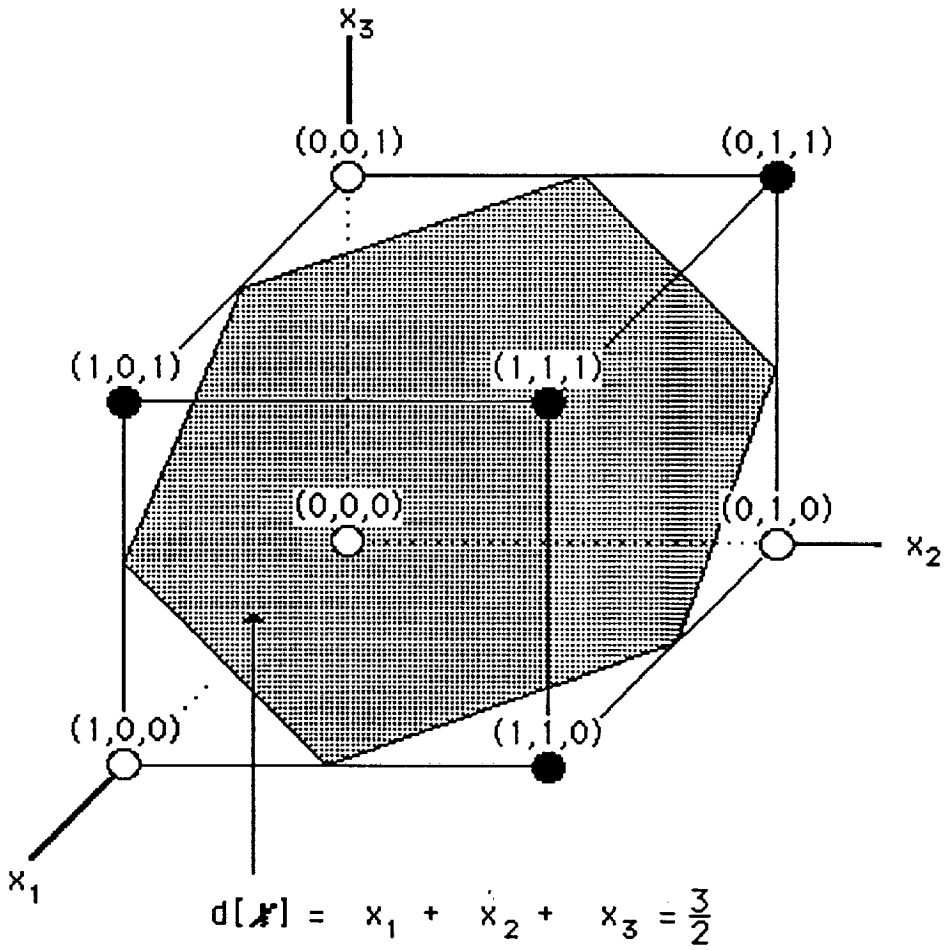
$$0<w_1,w_2,w_3<T. \qquad (*)$$

As a possible selection, choosing $w_1=w_2=w_3=1$ and $T=1.5$,

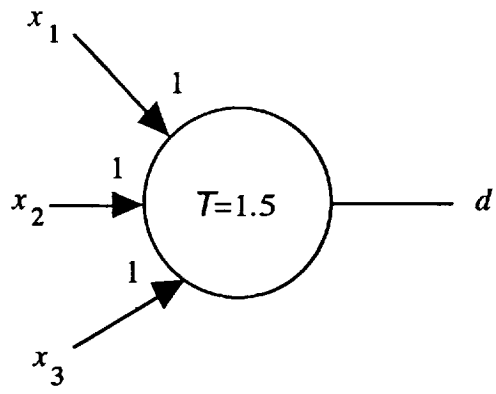
the hyperplane equation now becomes,

$$x_1 + x_2 + x_3 = 1.5.$$

It is important to note that there are an infinite number of w_i's and T's that satisfy (*). As an illustration, the following figure shows how the hyperplane equation separates one class of vectors from the other.



Also, a possible threshold logic unit realization for proper classification is shown as follows:



Although for this problem, the parameters(i.e., w_i's and T) are fairly easy to obtain by examining the inequalities from the truth table, other pattern classification problems that deal with many input features can be less trivial. For 9 inputs, 512 inequalities are needed. In general, n input features require 2^n inequalities. The solution to such a large set of inequalities is a challenging computation. Therefore, a simpler and more effective systematic method for solving for the parameters is desired. A possible method is presented in the following section.

5. A systematic method for obtaining the TLU's parameters

In order to find a simpler and more effective systematic method for obtaining a linear hyperplane that separates input patterns into different class, properties of linear separability are examined.

Given n binary valued features, there exists 2^n input patterns. If there exists a linear equation that separates patterns that correspond to outputs equal to zero(false nodes) to patterns that correspond to outputs equal to one(true nodes), then the patterns are linearly separable. This linear equation corresponds to a (n-1) dimensional hyperplane. From the n(binary valued) features, a boolean function f consisting of minimal sums-of-products(MSP) can be obtained by means of Quine-McCluskey tabular methods. Now, using the concept of lattices, for a hyperplane equation that separates the patterns to exist, it is necessary for the MSP function f to consist of each

variable x_i(x_i') to appear only in uncomplemented(complemented) form. If so, f is said to be unate. Unfortunately, the property of unatness is only a necessary condition for linear separability and not a sufficient one. Next, in order to obtain this hyperplane, it is necessary to find the two different sets of patterns that are the closest to each other. This can be achieved by obtaining the minimal true nodes and the maximal false nodes. The minimal true nodes are the set of patterns that constitute to the minterms of the MSP function f. The maximal false nodes are found by determining all false nodes with just one feature whose value is 0, then all false nodes with two features whose value is 0, and so on, leaving out all nodes smaller than the ones already selected. To determine whether or not f is linearly separable, and if it is to find an appropriate set of parameters, it is necessary to determine the coefficients of the hyperplane equation. This is accomplished by deriving and solving a system of pq inequalities(i.e., all combinations of false($\sum_{i=1}^{n} w_i x_i$) < true($\sum_{i=1}^{n} w_i x_i$)), corresponding to the p minimal true and q maximal false nodes. If the pq inequalities can be solved, then there exists a hyperplane that separates patterns correctly. For the example mentioned in section 4, the MSP form $f = x_1x_2+x_2x_3+x_1x_3$ and is unate. The minimal true nodes are (1,1,0),(0,1,1), and (1,0,1), and the maximal false ones are (0,0,1), (0,1,0), and (1,0,0). The system of inequalities yields all combinations of

$$\begin{cases} w_1 \\ w_2 \\ w_3 \end{cases} < \begin{cases} w_1 + w_2 \\ w_2 + w_3 \\ w_1 + w_3 \end{cases} \qquad (**)$$

and reducing the inequalities, we have

$0<w_1$

$0<w_2$

$0<w_3$.

If we let $w_1 = w_2 = w_3 = 1$, and substititing into (**) above, we obtain $1<2$ for all combinations of the system of inequalities. Now, since we want the threshold T to be located somewhere between

all combinations of false($\sum_{i=1}^{n} w_i x_i$) < true($\sum_{i=1}^{n} w_i x_i$), T=1.5 is a possible choice. This agrees with the results obtained from the example in section 4.

6. Introduction to multiple valued logic

Multivalued logic is a generalization of binary logic for an arbitrary number n of truth values,where n >2. The truth values (or degrees of truth) are usually chosen to be rational numbers between 0 and 1. The set T_n of truth values is usually defined as

$$T_n = \{ 0 = \frac{0}{n-1}, \frac{1}{n-1}, \ldots, \frac{n-1}{n-1} = 1 \}.$$

Many multivalued logics can be formulated depending on how the basic logic operations of disjunction, conjunction, negation, and implication are defined. For example Lukasiewicz logic uses the following definitions.

$$\bar{a} = 1 - a, \quad a \vee b = \max(a,b), \quad a \wedge b = \min(a,b),$$

$$\text{and} \quad a \rightarrow b = \min(1, 1 + b - a). \qquad (1)$$

When $n \rightarrow \infty$, we obtain an infinite valued logic where the truth values are all rational numbers in the interval [0,1] taken from the set T_∞. If we insist on taking all real numbers in the interval [0,1] rather than those from the set T_∞, we can obtain an alternative infinite-valued logic, usually denoted by L_1. This is also known as standard Lukasiewicz logic. However, these two infinite logics are essentially equivalent if one is one concerned with the tautologies they represent. There is a one-to-one correspondence (isomorphism) between set theory and logic because the set theoretic concepts of union, intersection, complement and inclusion correspond to the logical concepts of disjunction, conjunction, negation, and implication.

Fuzzy set theory is generalization of classical set theory where the membership value of an element in a set can take any value in the unit interval [0,1]. Given the isomorphism between set theory and logic, one can view fuzzy set theory based on max, min, and 1-a operators for union, intersection and complement as an infinite-valued standard Lukasiewicz logic L_1. Similarly, if we restrict the membership values that can occur in fuzzy set theory to the set T_n, then we obtain a

discrete fuzzy set theory which is essentially equivalent to multivalued logic. (One can make n as large as one wishes, depending on desired the accuracy of representation.)

For multi-criteria decision making and information fusion methods based on fuzzy set theory for pattern recognition and computer vision, the aggregation takes place in a hierarchical network, where the type of aggregation (conjunctive, disjunctive, etc) at each node is determined through a learning procedure. The learning procedures we have developed are based on gradient descent, and use training data to adjust the parameters of the nodes so that the sum of squared errors between the desired output and the actual output is minimized. Two powerful aspects of these learning methods are that they are capable of i) eliminating uninformative and unreliable criteria (i. e., pruning the decision tree), and ii) generating a set of decision rules automatically from the training data. However, since these algorithms are based on gradient descent methods, they can be slow and sometimes they may converge to local minima. Thus, alternative methods to synthesize such aggregation networks are highly desirable. One consequence of the equivalence between discrete fuzzy set theory and multivalued logic is that we can borrow concepts from multivalued logic to analyze and synthesize aggregation networks based on fuzzy set theory.

The problem of function minimization (to eliminate redundancy) and synthesis has been discussed in section 5 for the binary case (e.g. Quine-McCluskey). Several techniques are based on the concept of lattices. Algorithms to implement the resulting binary functions in terms of threshold logic units also exist in the literature (e. g. McNaughton). Hampson et al have derived some theoretical results for the case when the inputs are multi-valued, but the output is binary. Although a binary output may be sufficient for some pattern recognition applications, it is not general. Also, the authors do not suggest any methods for the construction of such networks. Recently there has been some interest in function minimization methods for multi-valued logic. The continued work proposed herein will draw from the existing literature and analyze both the theoretical and practical aspects of multi-valued logic function synthesis. These results will be then used to construct the decision functions in terms of fuzzy logic units for pattern recognition and computer vision. The advantages of the proposed methods are that

i) they are highly amenable to theoretical analysis,

ii) redundancy detection is handled naturally in the function minimization process, and

iii) the resulting network is always guaranteed to be globally optimal for the training data.

In fact, in the case of pattern recognition problems, the classification error on the training data will be zero.

7. Bibliography

1. G. J. Klir and T. A. Folger, *Fuzzy Sets, Uncertainty, and Information*, Englewood Cliffs, NJ: Prentice Hall, 1988.
2. S. E. Hampson and D. Volper, "Representing and Learning Boolean Functions of Multi-valued Features", *IEEE Transactions on Systems, Man and Cybernetics*, Vol. 20, No. 1, Jan/Feb 1990, pp. 67-79.
3. *Proceedings of the 21st International Symposium on Multiple-Valued Logic*, IEEE Computer Society Press, 1991.
4. Z. Kohavi, *Switching and Finite Automata*. New York: McGraw-hill, 1976.
5. E. J. McCluskey, *Logic Design Principles*. Englewood Cliffs, NJ: Prentice Hall, 1986.

Feature Calculation

Since we have experienced a delay in obtaining imagery from NASA, we have postponed much of the work on this task until that time when the data becomes available. As mentioned in the first quarter report, we have available numerous algorithms which we routinely use to generate features from digital images. These are ready, and will be run on the data when it arrives. The section on Acquisition of Images will detail our progress on generating our own "simulation" imagery.

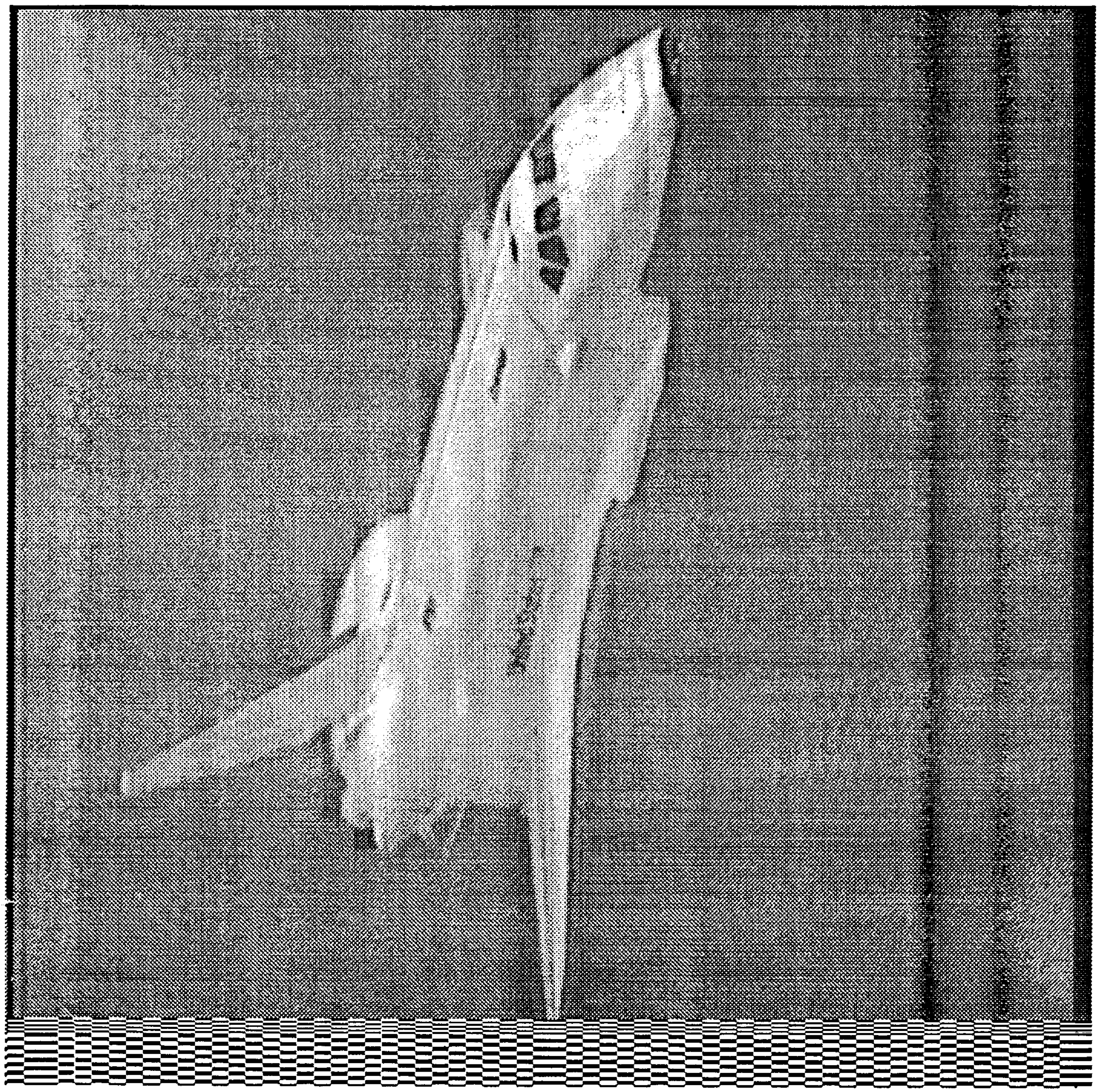
Calculation of Membership Functions

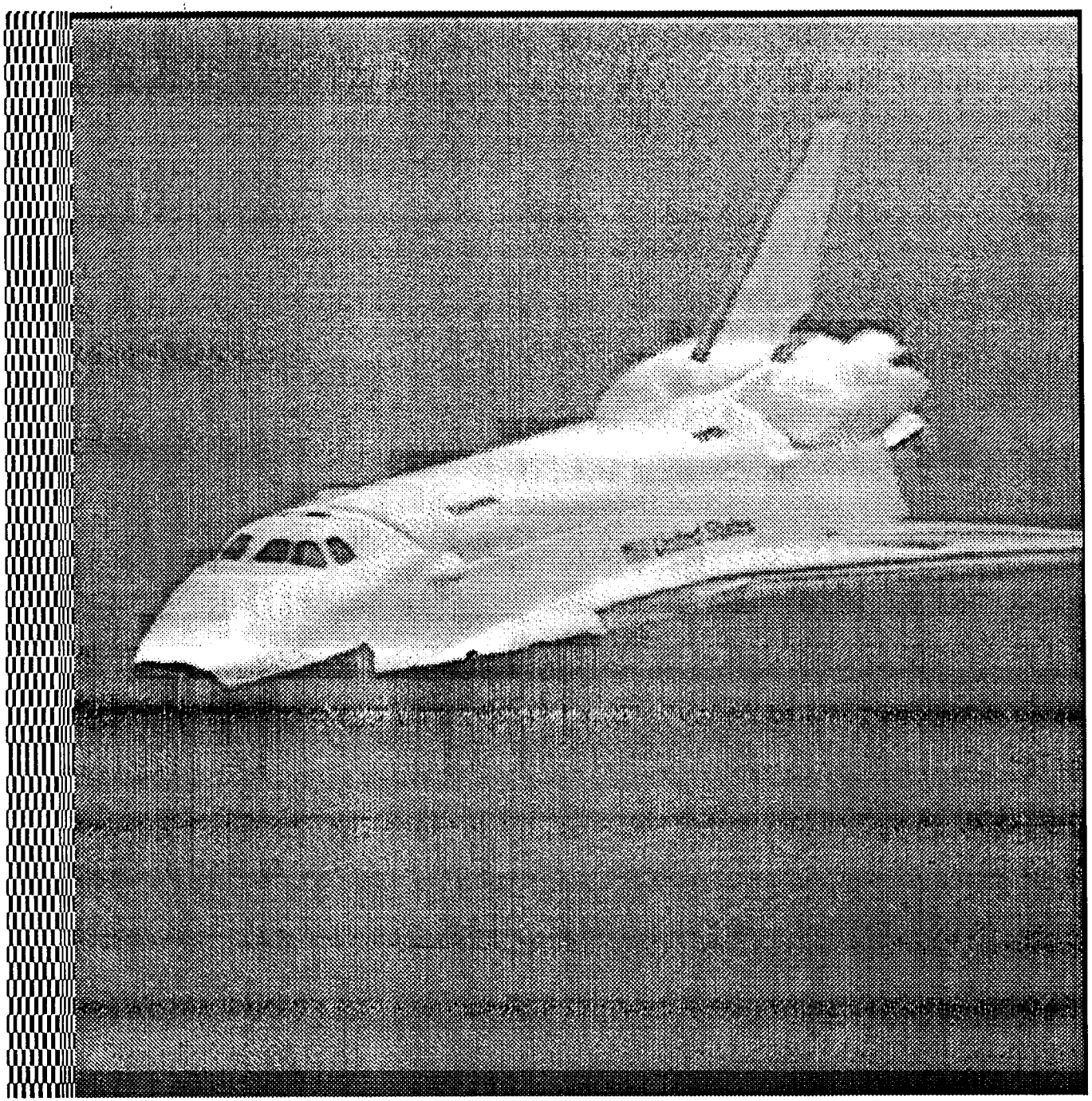
Our work in this area has progressed nicely. We have designed and implemented numerous algorithms to generate membership values from a set of training data using histograms, results of fuzzy clustering, and heuristic definitions. We have also made progress in the transformation of "probability density functions" into possibility distributions for use in assigning membership values to individual points. Since this task overlaps into the third quarter, we are going to postpone the complete write-up until the third quarter report. Hopefully, at that time, we will be able to supply a preprint of a paper describing our new results. We feel that that approach is the most profitable, since the paper will contain a concise statement and solution to the problem.

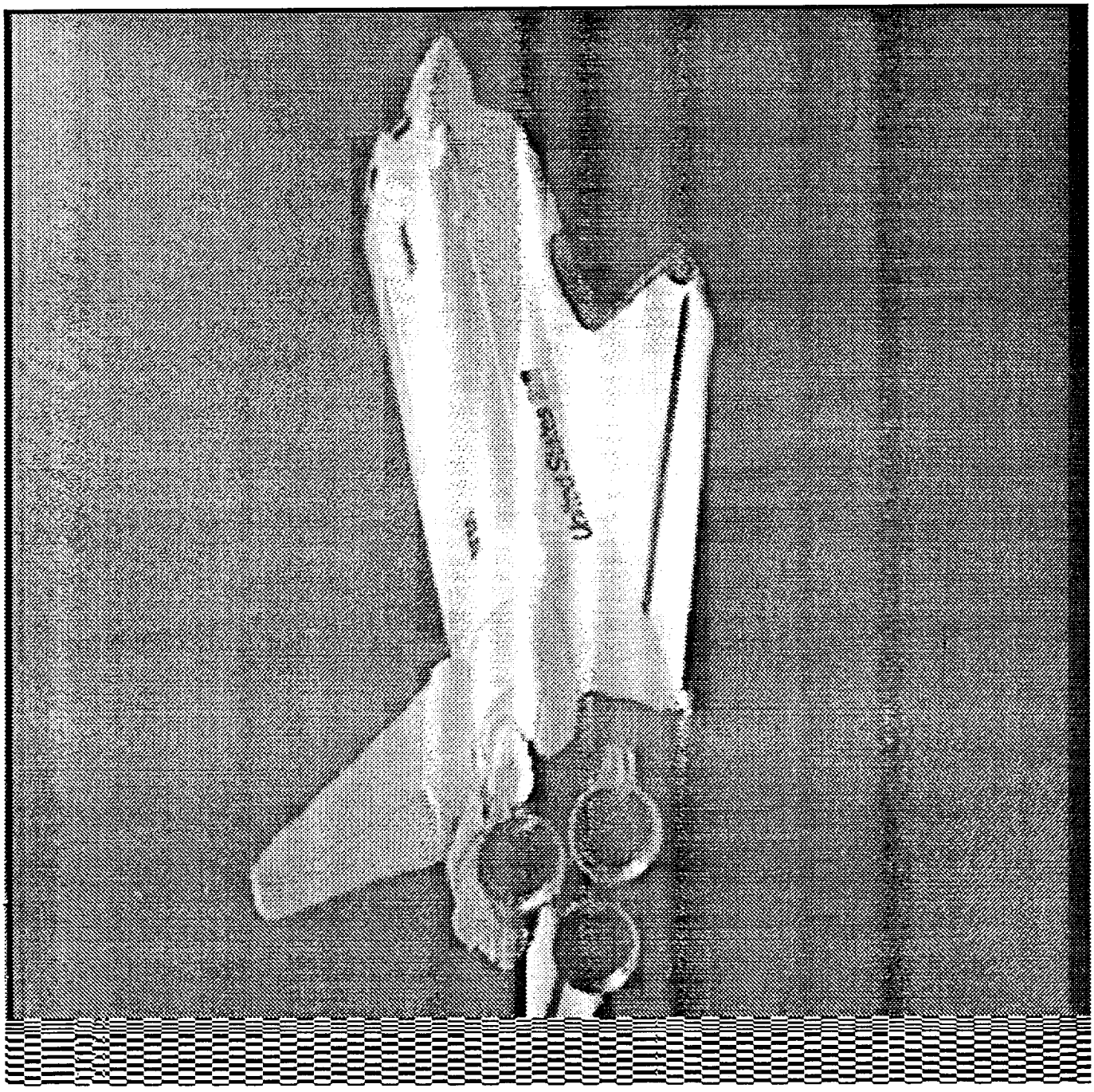
Acquisition of Images

As mentioned in the beginning of this report, we are waiting to receive imagery from NASA on which to test our algorithms. In the meantime, we have built a scale model of the shuttle, and built a mechanism to position this model at known orientations relative to the camera. We have begun to digitize images of this model to test some of the algorithms while we are waiting for the NASA pictures. We are including a few of these images both in hard copy form and on the accompanying tape.

Jaeger







Clustering for Curve and Surface Fitting

The best way to describe the new work in this task is to include a copy of a manuscript recently submitted by Dr. Krishnapuram and two of the graduate students supported by this contract to the 1992 IEEE Computer Vision/ Pattern Recognition Conference.

The title of the paper is:

"Quadric Shell Clustering Algorithms and Their Applications".

This represents the extension of the previously reported work to clustering edge data into general quadratic curves. We are also extending this approach to 3-Dimensional data sets(ie, surfaces).

Quadric Shell Clustering Algorithms and Their Applications

Raghu Krishnapuram, Hichem Frigui, and Olfa Nasraoui

Department of Electrical and Computer Engineering

University of Missouri, Columbia, MO 65211

ABSTRACT

In this paper, we introduce new hard and fuzzy clustering algorithms called the C Spherical Shells (CSS) algorithms and the C Quadric Shells (CQS) algorithms. The C Spherical Shells algorithms are specially designed to search for clusters that can be described by circular arcs, or more generally by shells of hyperspheres. The C Quadric Shells algorithms are expressly designed to seek clusters that can be described by segments of second-degree curves, or more generally by segments of shells of hyperquadrics. Most previous clustering algorithms assume that the clusters are "filled", i. e., they are not hollow. Such algorithms cannot cluster data that lie on shell-like subspaces of the feature space. The advantage of our CQS algorithms lies in the fact that they can be used to cluster mixtures of all types of hyperquadrics such as hyperspheres, hyperellipsoids, hyperparaboloids, hyperhyperboloids, and hypercylinders. We also introduce cluster validity measures for shell-like clusters, and show that the validity measure can be used to determine the number of clusters when this is not known. Several examples of clustering in the two-dimensional case are shown and their applications are suggested. These algorithms can easily outperform the traditional algorithms based on the Hough transform for boundary detection.

Acknowledgment:

This research was partially supported by NASA/JSC through subcontract No. 088 by the RICIS center at the University of Houston -Clear Lake (Cooperative Agreement No. NCC 9 - 16, Project No. SE. 42)

Summary

1. This paper is about partitioning n-dimensional data points which are assumed to lie on (possibly an unknown number of) hyperquadric surfaces into meaningful clusters. In other words, the clusters we deal with are described by shell-like subspaces of the original feature space.

2. We introduce new hard and fuzzy clustering algorithms called the C Spherical Shells (CSS) algorithms the C Quadric Shells (CQS) algorithms. The CSS algorithms search for clusters that can be described by circular arcs, or more generally by shells of hyperspheres. The CQS algorithms seek clusters that can be described by segments of second-degree curves, or more generally by segments of shells of hyperquadrics. We also introduce cluster validity measures for shell-like clusters, which can be used to determine the number of clusters when this is not known.

3. Most objective-function-based clustering algorithms in the literature consider only "filled" clusters, and hence they cannot be used when the clusters are hollow. The few shell clustering algorithms that have considered hollow clusters work only for clusters of specific shapes such as circles or ellipses in 2-D. The few existing algorithms are also implementationally complex since they require solving coupled nonlinear equations for the shell parameters. They do not perform well on partial shells either. Our algorithms do not involve nonlinear equations, i. e., they have solutions in closed-form. Thus, they are much faster. Moreover, they can be used to cluster mixtures of all types of hyperquadric shells such as hyperspheres, hyperellipsoids, hyperparaboloids, hyperhyperboloids, and hypercylinders. Our algorithms can easily outperform the traditional algorithms based on the Hough transform in boundary detection and other applications.

4. The proposed algorithms can be used for various applications such as boundary detection and pattern classification. They can also be used for surface fitting and description in 3-D (i. e., with range data). These algorithms can potentially lead to a more general class of algorithms that deal with shells of more complex types.

1. Introduction

Many clustering algorithms have been suggested and used in the literature to partition data into clusters. Clustering algorithms can be categorized into two classes, depending on whether a feature point belongs to just one cluster or to all C clusters, albeit to different degrees. These two classes are known as hard (crisp) and fuzzy algorithms respectively. There is an entire class of clustering algorithms in which an objective function based on a distance measure is iteratively minimized to obtain the final partition [1,2]. The distance measure chosen and the objective function being optimized depend on the geometric structure of the clusters. For example, the K-means algorithm, using the Euclidian distance, looks for hyperspherical clusters [3]. The Gustafson-Kessel algorithm uses a weighted Mahalanobis distance, and can detect hyperellipsoidal and hyperplanar clusters [1].

Until recently, it has been difficult to detect clusters that can be described by shell-like subspaces i.e., clusters that are not "filled" but are hollow. Dave's [4,5] Fuzzy C Shells (FCS) algorithm has proven to be successful in detecting clusters that can be described by circular arcs, or more generally by shells of hyperspheres. Several impressive examples involving two-dimensional data sets are given in [4,5]. Dave et al have also generalized this algorithm to the case of ellipsoidal shells [6,7] and this algorithm is known as the Fuzzy Adaptive C-Shells (FACS) algorithm. However the FCS and the FACS algorithms are implementationally complex since they involve the use of Newton's method to solve coupled nonlinear equations for the shell (prototype) parameters. Moreover, the performance of the FACS algorithm is not good for partial curves. A modification to the FCS algorithm has been suggested by Bezdek et al to reduce the computational burden [8].

In this paper, we propose new C Spherical Shells (CSS) algorithms that do not involve coupled nonlinear equations, i. e., they have solutions in closed-form. This makes our algorithm straightforward, and more importantly, computationally more attractive. In addition, we present a new set of hard and fuzzy clustering algorithms that generalize the Fuzzy C Shells and the

Adaptive Fuzzy C Shells algorithms. We call these algorithms the C Quadric Shells algorithms. They use an objective function based on a new distance measure and they seek clusters which can be described by segments of second degree-curves, or more generally by segments of shells of hyperquadrics. We also propose algorithms to determine the optimum number of clusters C, when this is not known. These algorithms involve minimizing a validity (performance) measure called the shell thickness. One major advantage of our CQS algorithms is that they are able to partition a composite mixture of different types of hyperquadric shells, whereas previous cluster-based and non-cluster-based algorithms apply to a specific type of hyperquadric. For example, Hough techniques to find analytical curves can be implemented efficiently only for specific types of curves [9]. This aspect is discussed further in Section 6. The other advantages of our approach are that they are computationally and implementationally simple, the memory and CPU time requirements are reasonable, and they do not require the a priori knowledge of the number of clusters present in the input data set.

Section 2 presents the hard and fuzzy versions of our C Spherical Shells algorithm. Section 3 introduces the hard and fuzzy versions of our C Quadric Shells (CQS) algorithm. Section 4 describes cluster validity measures and unsupervised algorithms which can be used to determine the optimum number of clusters when this is not known a priori. In Section 5 several examples of clustering using the proposed algorithms are presented and applications in computer vision and pattern recognition are suggested. Finally, section 6 gives the summary and conclusions.

2. The C Spherical Shells (CSS) Algorithms

In the case of C Spherical Shells algorithms, the assumption is that each cluster resembles a hyperspherical shell, or part thereof. Let x_j be a point in the feature space. The prototypes λ_i consist of two parameters (c_i, r_i), where c_i is the center of the hypersphere and r_i is the radius. We define the distance from x_j to a prototype $\lambda_i = (c_i, r_i)$ as

$$d_{Sij}^2 = d_S^2(x_j,\lambda_i) = (\| x_j - c_i \|^2 - r_i^2)^2. \qquad (1)$$

The subscript S in the above equation stands for "spherical". Note that the right hand side of (1), when equated to zero, also gives the equation of the hypersphere. In general, the closer x_j is to the specific hypersphere, the smaller the distance will be. Based on this distance measure, we now define the hard and fuzzy C Spherical Shells algorithms.

2.1 The Hard C Spherical Shells (HCSS) Algorithm

We define the objective function to be minimized in this case, as

$$J_S(L) = \sum_{i=1}^{K} \sum_{x_j \in \lambda_i} d_{Sij}^2, \tag{2}$$

where $L = (\lambda_1,...,\lambda_K)$, and K is the number of clusters. In order to minimize the objective function in (2), we rewrite the distance in (1) as

$$d_{Sij}^2 = p_i^T M_j p_i + v_j^T p_i + b_j,$$

where

$$b_j = (x_j^T x_j)^2, \qquad v_j = 2(x_j^T x_j) y_j, \qquad y_j = \begin{bmatrix} x_j \\ 1 \end{bmatrix},$$

$$M_j = y_j y_j^T, \quad \text{and} \quad p_i = \begin{bmatrix} -2 c_i \\ c_i^T c_i - r_i^2 \end{bmatrix}. \tag{3}$$

Therefore,

$$J_S(L) = \sum_{i=1}^{K} \sum_{x_j \in \lambda_i} (p_i^T M_j p_i + v_j^T p_i + b_j). \tag{4}$$

We may assume that the vectors p_i are independent of each other. Hence, the vectors p_i that minimize (4) must satisfy

$$\sum_{x_j \in \lambda_i} (2M_j p_i + v_j) = 0. \tag{5}$$

If we define

$$H_i = \sum_{x_j \in \lambda_i} M_j, \text{ and } w_i = \sum_{x_j \in \lambda_i} v_j, \tag{6}$$

from (5) we obtain

$$p_i = -\frac{1}{2} H_i^{-1} w_i \tag{7}$$

The resulting Hard C-Shells (HCS) algorithm is summarized below.

THE HARD C SPHERICAL SHELLS (HCSS) ALGORITHM:

Fix the number of clusters K;

Set iteration counter $l = 1$ and initialize the hard K-partition;

Repeat

Calculate $H_i^{(l)}$ and $w_i^{(l)}$ for each cluster using (6);

Compute $p_i^{(l)}$ for each cluster using (7);

Classify x_j into cluster λ_i if $d_{ij}^2 \le d_{kj}^2$, for all $k \neq i$;

Increment l;

Until ($\|p^{(l-1)} - p^{(l)}\| < \varepsilon$);

2.2 The Fuzzy C Spherical Shells (FCSS) Algorithm

For the fuzzy case, we minimize the following objective function:

$$J_S(L,U) = \sum_{i=1}^{K} \sum_{j=1}^{N} (\mu_{ij})^m d_{Sij}^2 . \tag{8}$$

In (8) N is the total number of feature vectors, and $U = [\mu_{ij}]$ is a $K \times N$ matrix called the fuzzy K-partition matrix [1] satisfying the following conditions:

$$\mu_{ij} \in [0,1] \text{ for all } i \text{ and } j, \ \sum_{i=1}^{K} \mu_{ij} = 1 \text{ for all } j, \text{ and } 0 < \sum_{j=1}^{N} \mu_{ij} < N \text{ for all } i.$$

μ_{ij} is the grade of membership of the feature point $\mathbf{x}_j$ in cluster λ_i, and $m \in [1,\infty)$ is a weighting exponent called the fuzzifier. As in the hard case, it is easy to show that the vectors p_i that minimize (8) are given by (7), where

$$H_i = \sum_{j=1}^{N} (\mu_{ij})^m M_j , \qquad w_i = \sum_{j=1}^{N} (\mu_{ij})^m v_j , \tag{9}$$

and v_j and M_j are given by (3). Using a proof similar to that of Bezdek's for the fuzzy C-means algorithm [1], it can be shown that the memberships will be updated according to

$$\mu_{ik} = \begin{cases} \dfrac{1}{\sum_{j=1}^{K} \left(\dfrac{d_{Sik}}{d_{Sjk}}\right)^{\frac{2}{m-1}}} & \text{if } I_k = \varnothing \\ 0 \quad i \notin I_k & \text{if } I_k \neq \varnothing \\ 1 \quad i \in I_k & \text{if } I_k \neq \varnothing \end{cases} \tag{10}$$

where $I_k = \{i \mid 1 \leq i \leq K, d^2_{Sik} = 0\}$. The resulting Fuzzy C Shells (FCSS) algorithm is summarized below.

THE FUZZY C SPHERICAL SHELLS (FCSS) ALGORITHM:

Fix the number of clusters K; fix m, $1 < m < \infty$;
Set iteration counter $l = 1$;
Initialize the fuzzy K-partition $U^{(0)}$;
Repeat
  Calculate $H_i^{(l)}$ and $w_i^{(l)}$ for each cluster λ_i using (9);
  Compute $p_i^{(l)}$ for each cluster λ_i using (7);
  Update $U^{(l)}$ using (10);
  Increment l;
Until $(\| U^{(l-1)} - U^{(l)} \| < \varepsilon)$;

Both the hard and fuzzy C-shells algorithms require the inversion of the matrix H_i. This is quite trivial when the feature space is two-dimensional or three-dimensional. In the hard case, the inverse will exist if there are at least $n+1$ non-collinear points in each cluster, where n is the dimensionality of the feature space. In the fuzzy case, theoretically the inverse will always exist as long as $N > n+1$ and the feature vectors are not collinear.

3. The C Quadric Shells (CQS) Algorithms

The C Spherical Shells algorithms can be generalized to include shells of (hyper)quadric surfaces, rather than just (hyper)spherical shells. We first present the two-dimensional case because it is easier to formulate. We then generalize the algorithm to the n-dimensional case.

Let $x_j = [x_{j1}, x_{j2}]$ be a point in the 2-D feature space. In the two-dimensional case, we assume that each cluster resembles a second-degree curve. Therefore, the prototypes β_i consist of six parameters $[a_{i1}, a_{i2}, \ldots, a_{i6}]$ which define the equation of the curve. We define the distance from x_j to a prototype β_i as:

$$d^2_{Qij} = d_Q{}^2(x_j,\beta_i) = (a_{i1}\, x_{j1}^2 + a_{i2}\, x_{j2}^2 + \sqrt{2}\, a_{i3}\, x_{j1}x_{j2} + a_{i4}\, x_{j1} + a_{i5}\, x_{j2} + a_{i6})^2. \qquad (11)$$

The subscript Q in the above equation stands for "quadric". The right hand side of (11), when set to zero, also represents the equation of the second-degree curve which the prototype represents. The coefficient of the $x_{j1}x_{j2}$ term in (11) is assumed to be $\sqrt{2}\, a_{i3}$ without loss of generality. This results in a simpler notation for a constraint that we will used later. Based on this distance measure, we now define the hard and fuzzy CQS algorithms.

3.1. The Hard C Quadric Shells Algorithm

In the hard case, we define the objective function to be minimized as

$$J_Q(L) = \sum_{i=1}^{K} \sum_{x_j \in \beta_i} d^2_{Qij}, \qquad (12)$$

where $L = (\beta_1,\ldots,\beta_K)$, and K is the number of clusters. In order to minimize the objective function in (12), we rewrite the distance in (11) as

$$d^2_{Qij} = d_Q{}^2(x_j,\beta_i) = (x_j^T A_i\, x_j + x_j^T\, v_i + b_i)^2 \qquad (13)$$

$$\text{where } A_i = \begin{bmatrix} a_{i1} & a_{i3}/\sqrt{2} \\ a_{i3}/\sqrt{2} & a_{i2} \end{bmatrix},\ v_i = \begin{bmatrix} a_{i4} \\ a_{i5} \end{bmatrix},\ \text{and}\ b_i = a_{i6}. \qquad (14)$$

Equation (13) can be rewritten as

$$d^2_{Qij} = p_i^T M_j\, p_i\ , \qquad (15\text{-a})$$

where the p_i represent the parameters of the prototypes of the clusters, and are given by

$$p_i^T = [p_{i1}^T \mid p_{i2}^T], \qquad p_{i1}^T = [a_{i1}, a_{i2}, a_{i3}], \quad \text{and} \quad p_{i2}^T = [a_{i4}, a_{i5}, a_{i6}]. \qquad (15\text{-b})$$

The M_j in (15-a) are given by

$$M_j = \begin{bmatrix} Q_j & R_j^T \\ R_j & S_j \end{bmatrix}, \tag{15-c}$$

where

$$Q_j = q_j\, q_j^T, \text{ with } q_j^T = [x_{j1}^2,\ x_{j2}^2,\ \sqrt{2}x_{j1}x_{j2}], \tag{16-a}$$

$$S_j = s_j\, s_j^T, \text{ with } s_j^T = [x_{j1},\ x_{j2},\ 1], \text{ and} \tag{16-b}$$

$$R_j = s_j\, q_j^T. \tag{16-c}$$

Using (15) and (16), (12) can be written as

$$J_Q(L) = \sum_{i=1}^{K} \sum_{x_j \in \beta_i} p_i^T M_j\, p_i\,. \tag{17}$$

$J_Q(L)$ is homogeneous with respect to p_i. Therefore, we need to constrain the problem in order to avoid the trivial solution. Some of the possibilities are:

(i) $\|p_i\|^2 = 1$

(ii) $b_i = 1$, and

(iii) $\mathrm{Tr}(A_i\, A_i^T) = \|p_{i1}\|^2 = 1$.

Constraint (ii) assumes that the curve does not pass through the origin. The last constraint has the advantage that the resulting distance measure is invariant to rigid transformations of the prototype [10]. However, if one is simply interested in clustering the data and not interested in obtaining invariant parameters of the clusters, one may use constraint (i). We found that constraint (iii) works best in practice.

While minimizing (17), we may assume that the vectors p_i are independent of each other. Hence, the objective is to minimize

$$J_Q(\beta_i) = \sum_{x_j \in \beta_i} p_i^T M_j\, p_i \qquad \text{subject to } \|p_{i1}\|^2 = 1. \tag{18}$$

If we define

$$F_i = \sum_{x_j \in \beta_i} Q_j, \qquad G_i = \sum_{x_j \in \beta_i} R_j, \qquad H_i = \sum_{x_j \in \beta_i} S_j, \text{ and} \tag{19}$$

$$W_i = \sum_{x_j \in \beta_i} M_j = \begin{bmatrix} F_i & G_i^T \\ G_i & H_i \end{bmatrix}, \text{ then}$$

using a Lagrange multiplier we can recast (18) as

$$J_Q(\beta_i, \lambda) = p_i^T W_i p_i - \lambda (\|p_{i1}\|^2 - 1). \tag{20}$$

Setting the gradient of $J_Q(\beta_i, \lambda)$ with respect to p_i equal to zero yields

$$W_i p_i = \lambda p_{i1}, \text{ and}$$

$$\begin{bmatrix} F_i & G_i^T \\ G_i & H_i \end{bmatrix} \begin{bmatrix} p_{i1} \\ p_{i2} \end{bmatrix} = \lambda \begin{bmatrix} p_{i1} \\ 0 \end{bmatrix}. \tag{21}$$

Equation (21) can be solved for p_{i1} and p_{i2}. The solution is given by

p_{i1} = eigenvector of $(F_i - G_i^T H_i^{-1} G_i)$ associated with the smallest eigenvalue. (22-a)

$$p_{i2} = - H_i^{-1} G_i p_{i1} \tag{22-b}$$

The resulting hard CQS algorithm is summarized below.

THE HARD C QUADRIC SHELLS (HCQS) ALGORITHM:

Fix the number of clusters K;

Set iteration counter $l = 1$;

Apply the HCSS algorithm until it converges to initialize the hard K-partition;

Repeat

Calculate $F_i^{(l)}$, $G_i^{(l)}$ and $H_i^{(l)}$ using (19) and (16);

Compute $p_i^{(l)}$ for each cluster using (22);

Classify x_j into cluster β_i if $d_{ij}^2 \leq d_{kj}^2$ for all $k \neq i$;

Increment l;

Until ($\| p^{(l-1)} - p^{(l)} \| < \varepsilon$).

Note that the HCSS algorithm has to be applied in order to get a good initial K-partition. Otherwise the performance of the HCQS algorithm is poor.

3.2. The Fuzzy C Quadric Shells algorithm

For the fuzzy case, we minimize the objective function

$$J_Q(L,U) = \sum_{i=1}^{K} \sum_{j=1}^{N} (\mu_{ij})^m \, d_{Qij}^2 , \tag{23}$$

where N is the total number of feature vectors and $U = [\, \mu_{ij} \,]$ is the $K \times N$ fuzzy K-partition matrix, as described in Section 2.2. As in the hard case, it is easy to show that the vectors p_i that minimize (23) subject to the constraint $\|p_{i1}\|^2 = 1$ are given by (22) where

$$F_i = \sum_{j=1}^{N} (\mu_{ij})^m \, Q_j ,$$

$$G_i = \sum_{j=1}^{N} (\mu_{ij})^m \, R_j , \qquad H_i = \sum_{j=1}^{N} (\mu_{ij})^m \, S_j , \tag{24}$$

and Q_j, R_j and S_j are as in (16).

Minimization with respect to the μ_{ij} can be done as before, if we follow Bezdek's theorem for the fuzzy C-means [1]. It can be shown that the memberships will be updated according to

$$\mu_{ik} = \begin{cases} \dfrac{1}{\sum_{j=1}^{K} \left(\dfrac{d_{Qik}}{d_{Qjk}} \right)^{\frac{2}{m-1}}} & \text{if } I_k = \varnothing \\ 0 \quad i \notin I_k & \text{if } I_k \neq \varnothing \\ 1 \quad i \in I_k & \text{if } I_k \neq \varnothing \end{cases} \tag{25}$$

where $I_k = \{ i \mid 1 \leq i \leq K, \, d_{Qik}^2 = 0 \}$.

The resulting FCQS algorithm is summarized at the end of section 3.3.

3.3. The n-dimensional case

Both the hard and the fuzzy C Quadric Shell algorithms can be extended to the n-dimensional case very easily. In this case, the distance measure is still in the form given by (13), where A_i is an $n \times n$ symmetric matrix given by

$$A_i = \begin{bmatrix} a_{i1} & a_{i(n+1)}/\sqrt{2} & . & . & . & a_{i(2n-1)}/\sqrt{2} \\ a_{i(n+1)}/\sqrt{2} & a_{i2} & . & . & . & . \\ . & . & . & . & . & . \\ . & . & . & . & . & . \\ . & . & . & . & . & a_{ir}/\sqrt{2} \\ a_{i(2n-1)}/\sqrt{2} & . & . & . & a_{ir}/\sqrt{2} & a_{in} \end{bmatrix}, \qquad (26\text{-a})$$

and v_i is an $n \times 1$ vector given by

$$v_i = \begin{bmatrix} a_{i(r+1)} \\ . \\ . \\ . \\ a_{i(r+n)} \end{bmatrix}, \text{ and } b_i = a_{i(r+n+1)}, \text{ where } r = \frac{n(n+1)}{2}. \qquad (26\text{-b})$$

This distance measure can again be written as (15-a), if the p_i are given by

$$p_i^T = [p_{i1}^T \mid p_{i2}^T], \qquad p_{i1}^T = [a_{i1}, a_{i2}, \ldots, a_{ir}], \qquad \text{and} \qquad p_{i2}^T = [a_{i(r+1)}, \ldots, a_{i(r+n+1)}].$$

Similarly (16) needs to be modified as:

$Q_j = q_j\, q_j^T$, where

$$q_j^T = [x_{j1}^2,\ x_{j2}^2, \ldots, x_{jn}^2, \sqrt{2}x_{j1}x_{j2}, \ldots, \sqrt{2}x_{jk}x_{jl}, \ldots, \sqrt{2}x_{j(n-1)}x_{jn}], \text{ and} \qquad (27\text{-a})$$

$$S_j = s_j\, s_j^T, \text{ where } s_j^T = [x_{j1}, x_{j2}, \ldots, x_{jn},\ 1]. \qquad (27\text{-b})$$

The Hard CQS algorithm remains the same if these new definitions are used. The fuzzy CQS algorithm is summarized below.

THE FUZZY C QUADRIC SHELLS (FCQS) ALGORITHM:

Fix the number of clusters K; fix m, $1 < m < \infty$;

Set iteration counter $l = 1$;

Initialize the fuzzy K-partition $U^{(o)}$ using the FCSS algorithm;

Repeat

Calculate $F_i^{(l)}$, $G_i^{(l)}$ and $H_i^{(l)}$ for each cluster β_i using (24) and (27);

Compute $p_i^{(l)}$ for each cluster β_i using (22);

Update $U^{(l)}$ using (25);

Increment l;

Until ($\| U^{(l-1)} - U^{(l)} \| < \varepsilon$);

The FCQS algorithm is somewhat sensitive to the initialization process. Hence the FCSS algorithm needs to be applied to obtain a reasonable initial fuzzy K-partition $U^{(0)}$.

3.4 The Modified C Quadric Shells Algorithm

The distance $d_Q^2(x_j,\beta_i)$ defined by (13) is highly nonlinear in nature. It is easy to show through simple examples that this distance is sensitive to the placement of x_j with respect to the prototype β_i. This does not cause problems if the clusters are well-defined quadric shells. However, if the clusters are ill-defined or if there is a lot of noise, the resulting estimates of the parameters of β_i and the memberships μ_{ij} can be significantly influenced by outliers. To alleviate this problem, one may use the shortest (perpendicular) distance (denoted by d_{Pij}^2) between the point x_j and the shell β_i. We now describe how this is achieved.

Let x_j be a point in the feature space. The distance measure d_{Pij}^2 is the minimum distance from the point x_j to the quadric curve β_i describing the cluster. Finding the d_{Pij}^2 can be formulated as:

$$d_{Pij}^2 = \min \|x_j - z\|^2 \text{ such that } (z^T A_i z + z^T v_i + b_i) = 0 \quad (28)$$

where z is a point lying on the quadric curve describing cluster β_i . Using a Lagrange multiplier λ, (28) reduces to minimizing $(\|x_j - z\|^2 - \lambda (z^T A_i z + z^T v_i + b_i))$ with respect to z and λ. This yields

$$2(x_j - z) + \lambda (2A_i z + v_i) = 0, \text{ and} \quad (29)$$

$$z^T A_i z + z^T v_i + b_i = 0 \quad (30)$$

Equation (29) can be solved for z as

$$z = \frac{1}{2}(I - \lambda A_i)^{-1} (\lambda v_i + 2x_j). \quad (31)$$

Substituting (31) in (30) yields an equation in λ which is a quartic (fourth-degree) equation in the 2-D case, and has at most four real roots λ_k, $1 \le k \le 4$. For higher dimensions the equation is of 6th degree or higher. Solving for the four roots in the 2-D case is quite straightforward if one uses the standard solution [11]. The resulting expressions are rather long and cumbersome, involving

nested square roots, and hence they are not presented here. For each real root λ_k so computed, we calculate the corresponding z_k using (31). Then, we compute d^2_{Pij} using

$$d^2_{Pij} = \min_k \|x_j - z_k\|^2 \tag{32}$$

One can formulate the FCQS algorithm using d^2_{Pij} as the underlying distance measure. In this case, the objective function to be minimized becomes

$$J_P(L,U) = \sum_{i=1}^{K} \sum_{j=1}^{N} (\mu_{ij})^m \, d^2_{Pij} \, . \tag{33}$$

Minimizing this function with respect to U yields

$$\mu_{ik} = \begin{cases} \dfrac{1}{\sum_{j=1}^{K} \left(\dfrac{d_{Pik}}{d_{Pjk}} \right)^{\frac{2}{m-1}}} & \text{if } I_k = \varnothing \\ 0 \quad i \notin I_k & \text{if } I_k \neq \varnothing \\ 1 \quad i \in I_k & \text{if } I_k \neq \varnothing \end{cases} \, . \tag{34}$$

However, minimizing (33) with respect to the parameters p_i results in coupled nonlinear equations with no closed-form solution. To overcome this problem, we may assume that we can obtain approximately the same solution by using (22), which will be true if all the feature points lie very close to the hyperquadric shells. This assumption leads to the following modified FCQS algorithm.

THE MODIFIED FUZZY C-QUADRIC SHELLS (MFCQS) ALGORITHM:

Fix the number of clusters K; fix m, $1 < m < \infty$;

Set iteration counter $l = 1$;

Initialize the fuzzy K-partition $U^{(o)}$ using the FCSS algorithm;

Repeat

Calculate $F_i^{(l)}$, $G_i^{(l)}$ and $H_i^{(l)}$ for each cluster β_i using (24) and (16);

Compute $p_i^{(l)}$ for each cluster β_i using (22);

compute d^2_{Pij} using (30),(31) and (32)

Update $U^{(l)}$ using (34);

Increment l;

Until ($\| U^{(l-1)} - U^{(l)} \| < \varepsilon$);

It is to be noted that this modified algorithm is easy to implement only in the 2-D case. In higher dimensions, solving for d_{Pij}^2 is not trivial. In practice, we found that in the 2-D case the modified FCQS algorithm converges much faster than the original version. This may be attributed to the fact that the membership assignment based on the perpendicular distance is more reasonable.

4. Determination of the Optimum Number of Clusters

The algorithms discussed in Sections 2 and 3 assume that the number of clusters K is known. This is indeed the case in many pattern recognition applications and some computer vision applications. When the number of clusters is unknown, one method to determine the optimal number of clusters is to perform clustering for a range of K values, and pick the K value for which a suitable performance measure is minimized (or maximized). For the fuzzy CQS algorithm, we define a new performance (or cluster validity) measure called the total fuzzy shell thickness as follows.

$$T_Q(K) = \sum_{i=1}^{K} \sum_{j=1}^{N} (\mu_{ij})^m d_{Pij}^2 , \tag{35}$$

which is also the objective function in (33). $T_Q(K)$ will be small if all points lie close to one of the K quadric shells. In the hard case, the total shell thickness measure becomes

$$T_Q(K) = \sum_{i=1}^{K} \sum_{x_j \in \beta_i} d_{Pij}^2 . \tag{36}$$

In the special case where all the quadric curves representing the clusters are (hyper)spheres, another validity measure called the total fuzzy average shell thickness for spherical shells may be defined as

$$T_S(K) = \sum_{i=1}^{K} \frac{\sum_{j=1}^{N} \mu_{ij}^m (\|x_j - c_i\| - r_i)^2}{\sum_{j=1}^{N} \mu_{ij}^m} \tag{37}$$

In the hard case, this becomes

$$T_S(K) = \sum_{i=1}^{K} \frac{1}{N_i} \sum_{x_j \in \lambda_i} (\|x_j - c_i\| - r_i)^2 \qquad (38)$$

where N_i is the number of points in cluster λ_i.

To find the optimum number of clusters when the FCQS algorithm is used, one can start with $K = 1$, and keep incrementing K while calculating $T_Q(K)$ after each run of the FCQS algorithm, and stop as soon as a knee point or a local minimum in the curve of $T_Q(K)$ is found (or K reaches K_{max}). This unsupervised algorithm is summarized below.

THE UNSUPERVISED FUZZY C QUADRIC SHELLS ALGORITHM:

Set $K = 1$; fix m, $1 < m < \infty$;
local_min _or_knee_point = false;
While $K <= Kmax$ **and** *local_min _or_knee_point* = false **do**
    Perform the FCQS algorithm with the number of clusters = K;
    Calculate $T_Q(K)$ as given by (35);
    If $T_Q(K\text{-}1)$ is a local minimum or a knee point **Then**
        local_min _or_knee_point = true;
        K_optimal = K-1;
    Else
        $K = K + 1$;
    End If
End While

Similar unsupervised algorithms can be designed with the HCSS, FCSS, and HCQS algorithms.

5. Experimental Results

In this section, we illustrate the algorithms presented in Sections 2, 3, and 4 through several examples. We present only results of two-dimensional data sets in this paper, even though the algorithms presented are applicable to feature spaces of any dimension. In general, we found that the hard CSS and CQS algorithms are about an order of magnitude faster than their fuzzy

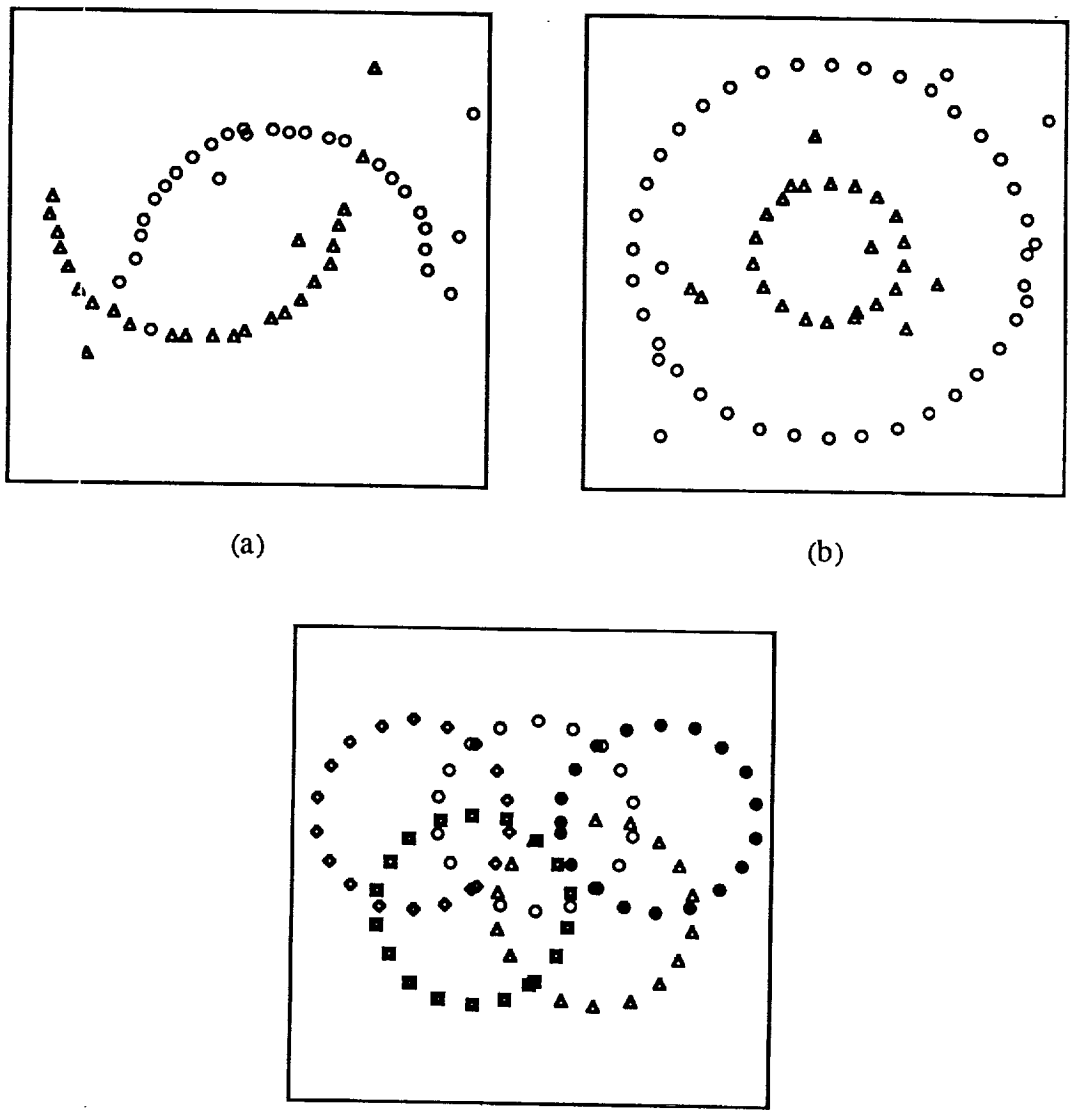
counterparts, but they perform well only when the clusters are not highly entangled. Thus, the hard versions are not very robust, and hence we do not present their results in this paper.

We first show the results of the FCSS algorithm. In these examples, the number of clusters is assumed to be known, although an unsupervised algorithm to find the optimum number of clusters can easily be devised using the performance measure in (37). In all the examples shown, the FCSS algorithm was applied with the fuzzifier $m = 5$. Smaller values did not yield good results. This may be because we initialize the fuzzy partition matrix U with the fuzzy C means algorithm [1] which does not yield a good partition of the clusters, particularly in the case of overlapping or concentric circles. By making the partitioning as fuzzy as possible, it is possible to disentangle the intertwined clusters from each other using the FCSS algorithm. The data sets in images of size 200×200 shown in Figure 1 were artificially generated, and had between 50 and 200 feature points. Uniformly distributed noise with an interval of 3 was added to the feature point locations so that they do not always lie exactly on the ideal circles. In addition, noise points were added at random locations to some of the data sets.

Figure 1(a) shows the result of clustering two semicircles contaminated by noise. This example shows that the algorithm is successful even when only parts of circles are present. The second example in Figure 1(b) consists of two concentric circles contaminated by a few noise points. This is an example where conventional clustering methods fail miserably. As seen in Figure 1(b), the two concentric circles are correctly classified, and the noise points are assigned to the closest cluster. Figure 1(c) shows the clustering of five sparsely sampled overlapping circles. This is a very difficult case, because the circles are truly entangled, and the initial partition is quite wrong. The CPU time required on a Sun 4 workstation to run the FCSS algorithm is typically on the order of 1s.

We next present the results of the unsupervised MFCQS algorithm. In all the examples shown in this paper, the initial fuzzy K-partition was obtained as follows. The Fuzzy C-Means

algorithm was first applied with the fuzzifier $m = 2$ for five iterations. The resulting fuzzy partition U was quite poor at this point. Therefore, this was followed by the application of the FCSS algorithm with $m = 2$. After the FCSS algorithm converged, the MFCQS algorithm was applied with $m = 2$.



(a) (b)

(c)

Figure 1: Examples of clustering using the Fuzzy C Spherical Shells algorithm. (a) two semi-circles with noise, (b) two concentric circles, and (c) five entangled sparsely sampled circles.

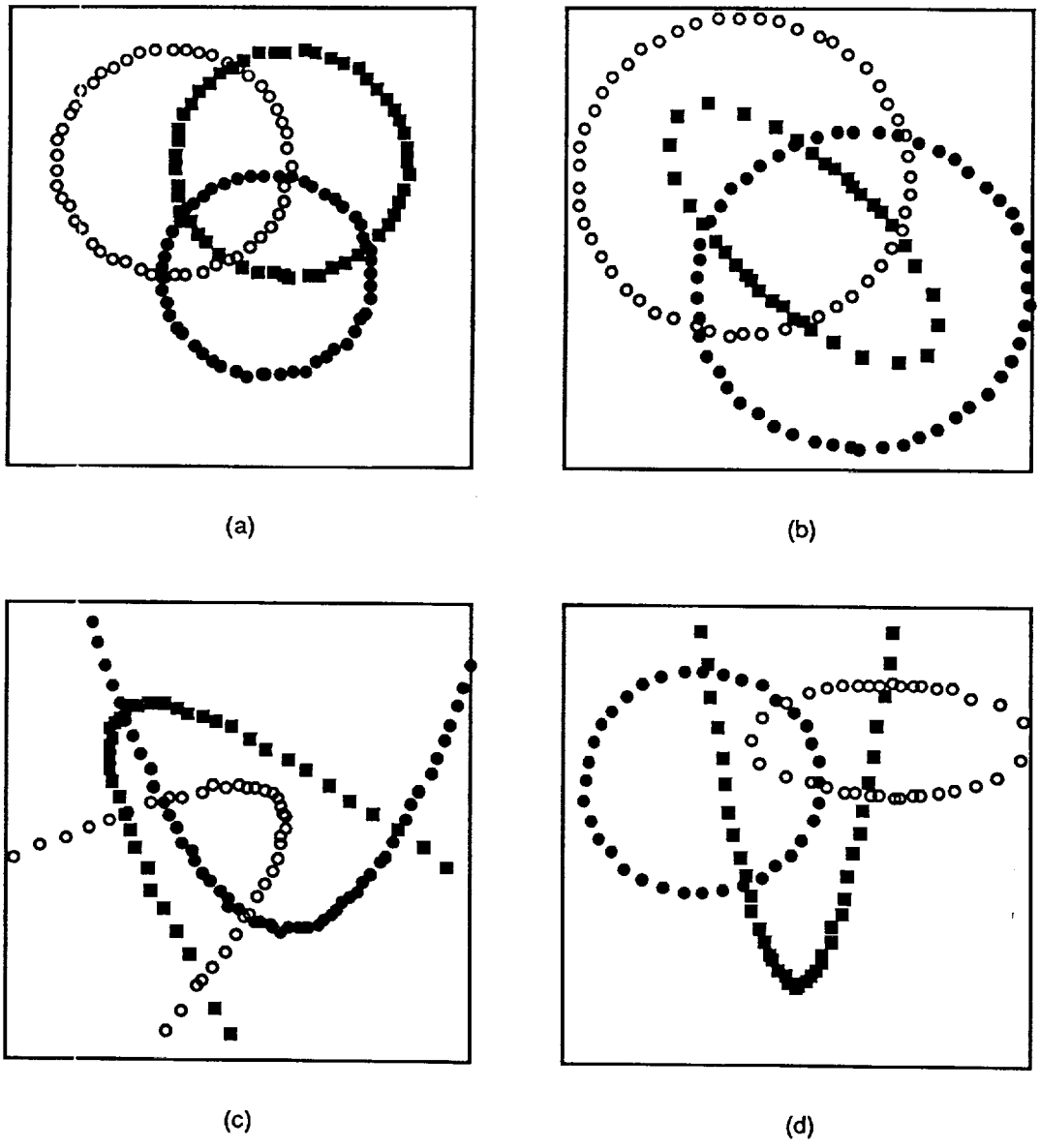


Figure 2: Examples of clustering using the modified fuzzy C Quadric Shells algorithm. (a) three overlapping circles, (b) an ellipse enclosed by two overlapping circles, (c) three parabolas with different orientations, and (d) a mixture of three types of quadrics: a circle, an ellipse and a parabola.

Figure 2 shows some examples that are typical of boundary detection problems. The data sets in images of size 200×200 were artificially generated, and had between 50 and 200 points.

Uniformly distributed noise with an interval of 3 to 5 was added to the feature point locations so that they do not always lie on ideal quadric curves. Example 1 consists of three overlapping circles, Example 2 shows two overlapping circles enclosing another ellipse, Example 3 shows three differently oriented parabolas, and Example 4 shows three different types of quadric curves: a parabola, a circle and an ellipse, all in the same data set. In all cases the clusters criss-cross one another, and conventional methods cannot separate them. The plot of the total fuzzy shell thickness vs the number of clusters for these four examples is shown in Figure 3. In every case, the knee point or the local minimum is clearly defined, and picking the optimum K is quite simple. The MFCQS algorithm clusters all these data sets successfully, and the results are excellent.

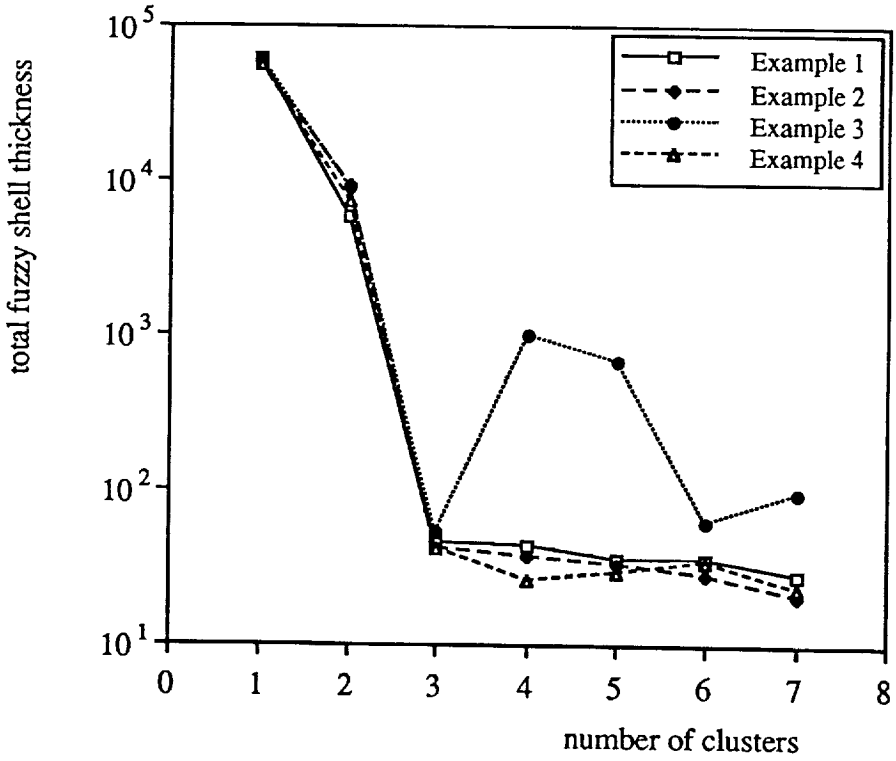


Figure 3: The plot of the total fuzzy thickness vs the number of clusters.

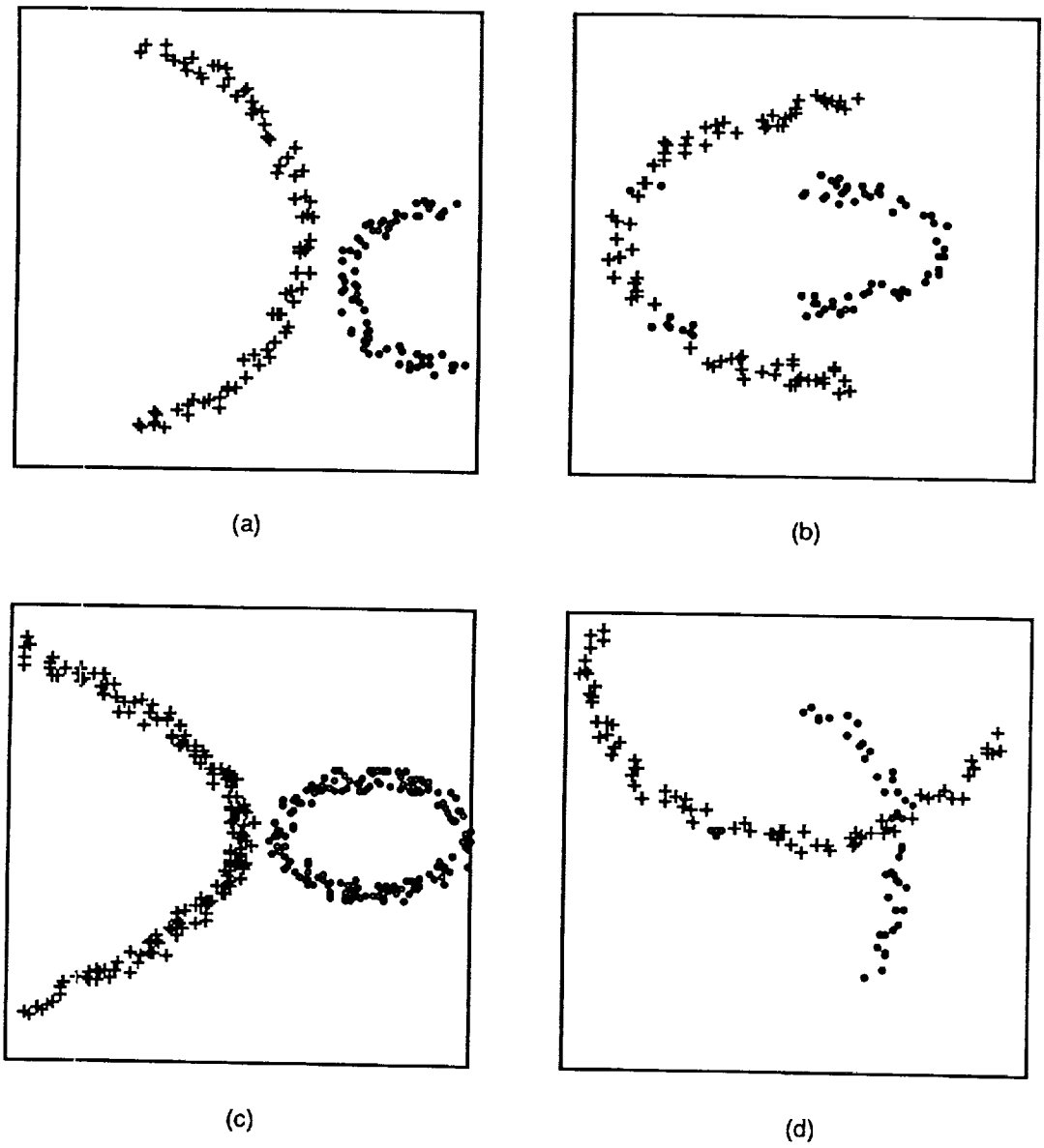


Figure 4: Examples of unsupervised modified fuzzy quadric shell clustering. (a) two (partial) circular clusters, (b) two elliptical clusters, (c) a parabolic cluster and an elliptic cluster, and (d) two crossing elliptical clusters.

Figure 4 shows several situations that are more typical of pattern recognition problems. These data sets in images of size 200×200 were also artificially generated, and each example has between 200 and 350 points. Uniformly distributed noise with an interval of 10 to 15 was added to

the feature point locations so that they do not lie on ideal quadric curves. Figure 4(a) (Example 5) shows the result of the unsupervised MFCQS clustering of two circular clusters with different radii. Fig. 4(b) (Example 6) shows the result of the unsupervised MFCQS clustering of two semi-elliptical clusters with very different major and minor axes. Fig. 4(c) (Example 7) shows the result obtained when the unsupervised MFCQS algorithm was used on a parabolic cluster and an elliptic cluster. Finally, Fig. 4(d) (Example 8) shows the unsupervised MFCQS clustering of two crossing elliptic clusters. These examples show that the MFCQS algorithm is effective even when the clusters are very different in size and when only partial curves are present. The plot of the total fuzzy shell thickness against the number of clusters is shown in Figure 5. Here again, it is vary easy to pick the knee point and the optimum number of clusters.

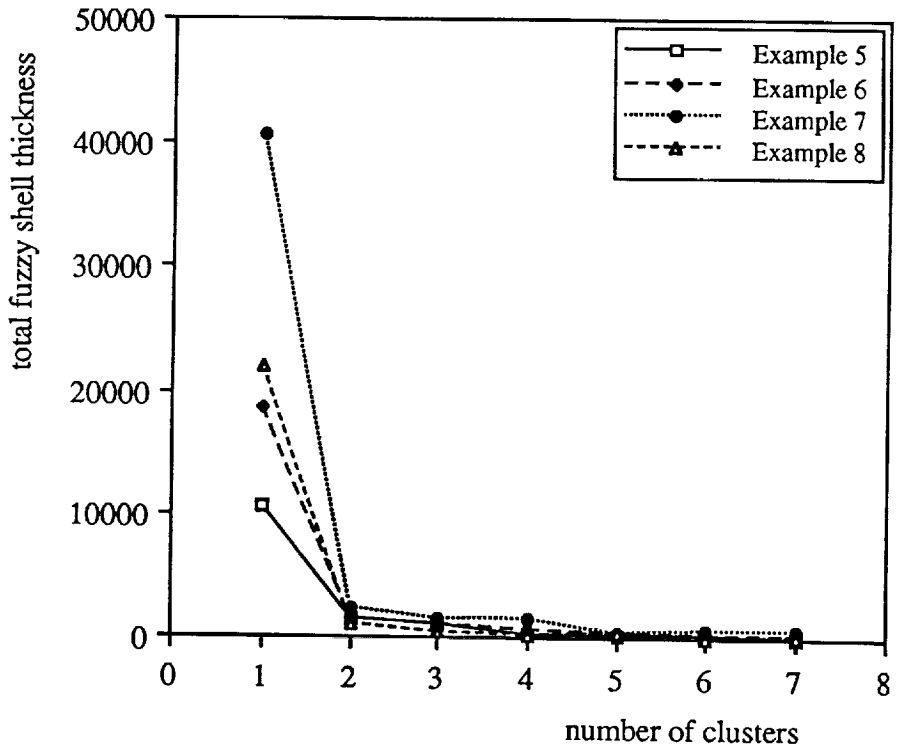


Figure 5: The plot of the total fuzzy shell thickness against the number of clusters.

In all the examples shown, the MFCQS algorithm performed successfully, yielding the correct final partition of the data set. It performed well in the presence of quadric clusters of the

same type or of different types and sizes. It also performed well with clusters that represent partial quadric curves of the same type or different types and sizes. The MFCQS algorithm typically converged in less than 20 iterations. The CPU time required on a Sun 4 workstation to run the MFCQS algorithm was typically on the order of 15s. This is very reasonable considering the complexity of the problems.

6. Conclusions

In this paper, we introduced new hard and fuzzy clustering algorithms called the C Spherical Shells (CSS) and C Quadric Shells (CQS) algorithms. These algorithms are specifically designed to seek clusters that can be described by segments of second-degree curves, or more generally by segments of shells of hyperquadrics. These algorithms can potentially lead to a more general class of algorithms that deal with shells of more complex types. Most objective-function-based clustering algorithms in the literature consider only filled clusters, and hence they cannot be used when the clusters are hollow. The few shell clustering algorithms that have considered hollow clusters work only for clusters of specific shapes such as circles or ellipses. The CSS algorithms are excellent for the detection of circular boundaries or clusters. The advantage of our CQS algorithms lies in the fact that they can be used to cluster mixtures of all types of hyperquadric shells such as hyperspheres, hyperellipsoids, hyperparaboloids, hyperhyperboloids, and hypercylinders. The examples shown in Section 5 of clustering in the two-dimensional case illustrate the superior performance of the proposed algorithms. The hard versions do not perform as well when the clusters are highly entangled, which shows the benefits of the fuzzy approach.

The proposed algorithms also have several advantages over the generalized Hough transform (GHT) methods that have been traditionally used to detect shapes of known descriptions. One disadvantage of the GHT approach is that one needs to use a different GHT for each type of curve. For example, one needs a GHT for circles, another for ellipses, and yet another for parabolas. Although one could devise a GHT that can cover all types of second-degree curves

(or hyperquadrics), the dimensionality of the resulting parameter space (six in the case of second-degree curves in 2-D) will be very large, and the resulting GHT would be computationally very expensive. The memory requirements can also be prohibitive[12]. The speed of the GHT can be improved only if we make certain assumptions about the curve, (for example, if the curve is an ellipse etc) and if the gradient information is available. Also, our algorithms work well even when the edge points are somewhat scattered around the ideal curve (or hypersurface), which causes bin splitting in the GHT. Our algorithms can locate small shell segments much better. Small peaks in the GHT are lost in the bias[13], and selecting a suitable threshold is difficult.

7. References

1. J. C. Bezdek, *Pattern Recognition with Fuzzy Objective Function Algorithms*, Ch. 3, Plenum Press, New York, 1981.

2. A. K. Jain and R. C. Dubes, *Algorithms for Clustering Data*, Prentice Hall, Englewood Cliffs, NJ, 07632, 1988.

3. J. Tou and R. C. Gonzalez, *Pattern Recognition*, Addison Wesley, Reading, MA, 1974.

4. R. N. Dave and S. K. Bhamidipati, "Application of the fuzzy-shell clustering algorithm to recognize circular shapes in digital images", *Proceedings of the International Fuzzy Systems Association Congress*, pp. 238-241, Seattle 1989.

5. R. N. Dave, "Fuzzy-shell clustering and applications to circle detection in digital images", *International Journal of General Systems*, vol. 16, pp. 343-355, 1990.

6. R. N. Dave and K. J. Patel, "Fuzzy ellipsoidal-shell clustering algorithm and detection of ellipsoidal shapes", *Proceedings of the SPIE Conference on Intelligent Robots and Computer Vision IX: Algorithms and Techniques*, pp. 320-333, Boston, Nov. 1990.

7. R. N. Dave, "Adaptive C-shells clustering", *Proceedings of the North American Fuzzy Information Processing Society Workshop*, pp. 195-199, Columbia, Missouri, 1991.

8. J. C. Bezdek and R. J. Hathaway, "Accelerating convergence of the Fuzzy C-Shells clustering algorithms", *Proceedings of the International Fuzzy Systems Association Congress*, Volume on *Mathematics*, pp. 12-15, Brussels, July 1991.

9. J. Illingworth and J. Kittler, "A Survey of Hough Transforms", *Computer Vision, Graphics and Image Processing*, pp. 87-116, 1988.

10. O. D. Faugeras and M. Hebert, *Techniques for 3-D Machine Perception*, A. Rosenfeld (Editor), pp. 13-51, Elsevier Science Publishers B. V. (North-Holland), 1986.

11. *Standard Mathematical Tables*, 21st Edition, S. M. Selby, Ed., The Chemical Rubber Co., Cleveland, OH 44128, 1973.

12. V. Milenkovic, "Multiple Resolution Search Techniques for the Hough Transform in High Dimensional Parameter Spaces", in *Techniques for 3-D Machine Perception*, A. Rosenfeld, Ed.,Elsevier Science Publishers, North Holland, 1986, pp. 231-255.

13. C. M. Brown, "Inherent Bias and Noise in Hough Transform", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 5, No. 5, Sept. 1983, pp. 493-505.